MASTER EN SCIENCES INFORMATIQUES

Spécification et implémentation d'un modèle d'information de bureau

Dachouffe, Nadine

Award date:
1986

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix - NAMUR

Institut d'informatique

SPECIFICATION ET IMPLEMENTATION D'UN MODELE D'INFORMATION DE BUREAU

Mémoire présenté par

Nadine DACHOUFFE

en vue de l'obtention du titre de Licencié et Maitre en Informatique.

Année académique 1985 - 1986

REMERCIEMENTS.

Je remercie Monsieur Lesuisse, promoteur de ce mémoire, pour l'intérêt qu'il a apporté à ce travail et pour l'aide et les conseils qu'il m'a prodigués tout au long de la réalisation et de la rédaction du mémoire.

Je remercie également Madame Mossiat-Ippersiel, secrétaire à l'institut d'informatique, de m'avoir permis de réaliser l'interview qui a servi de base à ce mémoire.

Je remercie Monsieur Bodart et son équipe, en particulier Monsieur Leheureux et Monsieur Clantin, pour l'aide qu'ils m'ont apportée dans la compréhension et l'utilisation du système IDA.

Je remercie également Monsieur Paquet de m'avoir fait profiter de son expérience dans l'utilisation du gestionnaire d'écran VAX-11 TDMS.

CHAPITRE 1: INTRODUCTION

CHAPITRE 2: LES MODELES EXISTANTS

CHAPITRE 3: MIB: UN MODELE D'INFORMATION DE BUREAU

CONCLUSION

BIBLIOGRAPHIE

CHAPITRE 1: INTRODUCTION

1 LE CONTEXTE DU PROBLEME ABORDE

Les systèmes d'information sont maintenant utilisés dans beaucoup de domaines d'application, et parallèlement, divers types de systèmes ont émergé et de nombreuses approches méthodologiques ont été développées. L'automatisation du travail de bureau est un des domaines d'application des systèmes d'information qui croit rapidement. Beaucoup d'efforts ont été réalisés, ces dernières années, dans le développement de nouvelles approches de la conception de systèmes d'information de bureau, considérant les aspects originaux de l'environnement bureautique.

L'identification et la prise en compte de ces différentes facettes et aspects originaux du bureau constituent une question fondamentale. Ces caractéristiques originales sont particulièrement utiles dans les premières phases de conception appelées phases "logiques" ou "conceptuelles". Ces caractéristiques originales par rapport aux systèmes d'information conventionnels concernent notamment:

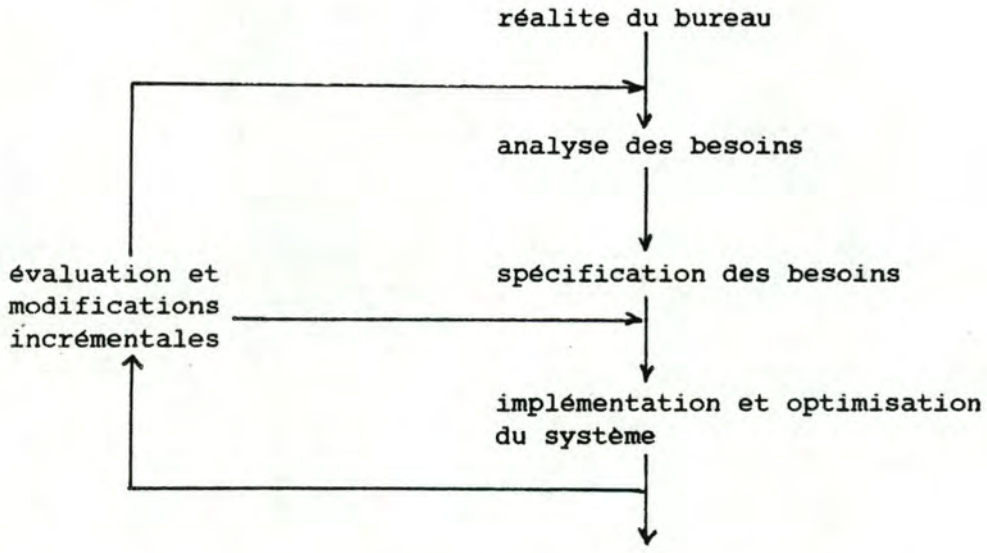
- les données de bureau:
  Les types de données utilisés dans les systèmes d'information conventionnels, tels que le caractère, le string, les données numériques, ne sont pas suffisants dans l'environnement de bureau. Il faudrait une sémantique plus riche, considérant tous les éléments du bureau, tant organisationnels qu'informationnels, avec leurs caractéristiques particulières.

- les tâches de bureau:
  Le travail de bureau consiste en un grand nombre de tâches opérationnelles et décisionnelles - en général, faiblement structurées - avec beaucoup d'anomalies possibles, ce qui rend l'analyse du travail de bureau très complexe.

- l'interconnexion des éléments:
  Les éléments nécessaires pour exécuter le travail de bureau sont distribués parmi plusieurs travailleurs de bureau et peuvent ainsi être localisés de façon externe à l'environnement de bureau. Donc, une des principales fonctions du bureau est la communication parmi les travailleurs de bureau et avec le monde extérieur.

L'analyse conceptuelle d'un système d'information de bureau constitue une tâche complexe. En fait, dans la conception d'un système d'information de bureau, il est d'abord nécessaire de définir le bureau et ses objectifs afin de comprendre quel travail on exécute globalement et comment le système à concevoir affectera le travail. L'analyse du travail de bureau qui devra être exécutée pour avoir une telle connaissance est également complexe, cela étant dû à la nature du travail de bureau lui-même.

Les phases de conception d'un système d'information de bureau sont similaires aux phases que l'on trouve dans les méthodologies de systèmes d'information conventionnels pour la conception de base de données et de systèmes d'information:



Chaque phase de conception d'un système d'information de bureau montre plusieurs différences avec la conception des systèmes d'information conventionnels, puisque l'on prend en compte les caractéristiques originales du bureau. Ces différences sont particulièrement nombreuses et significatives dans les premières phases de la conception, à savoir l'analyse et la spécification des besoins.

2 L'OBJECTIF DU MEMOIRE

L'objectif de ce mémoire est de fournir un environnement sous forme de modèles et d'outils automatisés, favorable à la construction d'un schéma conceptuel, c'est-à-dire une description

- des informations
- des traitements qui y sont associés, et
- des enchainements de ces traitements

qu'il conviendra de représenter dans un système d'information bureautique pour satisfaire les besoins exprimés dans le bureau. Le schéma conceptuel construit devra être:

- communicable: précis et standard;
- complet: tous les détails s'y trouvent;
- cohérent: sans ambiguité ni contradiction;
- conforme aux besoins: garantissant que le système d'information bureautique réalisé produira bien les résultats attendus par l'organisation.

La conception d'un système d'information de bureau est un gros travail: nous ne le réaliserons donc pas complètement, ce qui nous empêchera de proposer une approche méthodologique complète. Nous mettrons particulièrement l'accent sur les phases d'analyse et de spécification des besoins. Cette spécification des besoins s'appuyera sur l'élaboration de différents modèles.

Pour ce qui concerne la phase d'implémentation du système, nous nous proposons, non pas d'écrire complètement un système logiciel pour implémenter les modèles élaborés dans la phase de spécification, mais nous implémenterons notre modèle détaillé à partir d'un système conventionnel plus générique: le système IDA. Ce choix se justifie pour les raisons suivantes:

- cela permet de récupérer toute une technologie existante: langage de spécification, outils documentaires, outils de simulation;
- cela répond à une philosophie de base consistant à utiliser un outil commun générique sur lequel on greffe différents interfaces de façon à implémenter divers systèmes plus détaillés et plus spécifiques.

3 LES RESULTATS ATTENDUS

Les résultats attendus de la création d'un modèle de bureau sont de 3 types:

- spécification;
- documentation;
- simulation

3.1 LA SPECIFICATION

Jusqu'il y a peu, la spécification d'un système de bureau plus ou moins automatisé était restée une tâche manuelle: une équipe de conception communique avec l'utilisateur pour créer et intégrer les nombreux modules du système, en utilisant le crayon, le papier et peut-être une approche de conception. Cela implique plusieurs problèmes: erreurs humaines, erreurs de communication, incompatibilité des interfaces des différents modules, redondance, incomplétude, incohérence. De plus, ces erreurs de spécification restent souvent non détectées jusqu'à l'implémentation et les tests.

Aussi, le premier but de la création d'un modèle de bureau doit-il être d'aider le concepteur à éviter le plus grand nombre possible d'erreurs.

Pourquoi écrire un nouveau modèle alors qu'il existe déjà des systèmes de conception de systèmes d'information ?

La conception d'un système d'information de bureau est bien sûr un cas particulier de la conception d'un système d'information conventionnel; les systèmes existants peuvent donc être utilisés. Mais l'objectif de la création d'un système de conception spécifique au bureau est d'exploiter les aspects particuliers du bureau (les objets caractéristiques, les opérations sur ces objets, l'organisation du bureau, l'environnement) afin de faciliter et accélérer la définition et l'analyse des tâches de bureau.

Dans cette tâche de conception, on peut aider l'utilisateur en lui fournissant en plus du modèle :

- un langage pour définir le schéma conceptuel du système d'information. Dans notre cas, ce langage devra comprendre des expressions de haut niveau (tant en ce qui concerne les données que les opérations) pour la description du bureau.
- une méthode proposant au concepteur un ordre de définition des différents éléments du schéma conceptuel. Comme nous n'implémenterons pas entièrement le système d'information de bureau proposé dans ce mémoire, cette approche méthodologique complète ne sera pas réalisée ici.

3.2 LA DOCUMENTATION

La documentation peut se présenter sous 2 formes:
- l'extraction de certaines descriptions à partir des spécifications décrites;
- l'interrogation du système d'information.

3.2.1 L' EXTRACTION DE DESCRIPTIONS

Plusieurs personnes sont souvent chargées de la conception d'un même système d'information, chacune devant être informée durant tout le processus de conception.

De plus, tout au long de la vie du système d'information, non seulement l'analyste ou le concepteur du système, mais aussi les personnes qui en ont demandé l'implémentation, devront être informés afin de valider le système, suggérer des modifications, identifier les inconsistances, ...

C'est pourquoi, on devrait pouvoir utiliser le modèle et le langage associé pour obtenir de la documentation.

A partir du modèle, on devrait pouvoir éditer des rapports documentaires concernant entièrement ou partiellement la structure du système. Décrivons les différents rapports utiles dans le cadre de la bureautique.

Un premier rapport concernerait la restitution des spécifications décrites. Cette fonction permettrait de visualiser le contenu du système.
Exemple: le concepteur du système d'information de bureau a décrit, dans le langage du modèle, un formulaire. Pour ce, il a donné:
- le nom du formulaire;
- une information permettant de l'identifier;
- une description complète du contenu du formulaire;
- l'origine et les destinations successives du formulaire;
- les manipulations de ce formulaire par différents traitements.

A tout moment, le concepteur ou une autre personne devrait pouvoir visualiser toute cette information introduite, en donnant le nom et l'identifiant du formulaire, afin d'en prendre connaissance et éventuellement, vérifier que cette description est correcte, la modifier, ...

Dans les bureaux, on envisage des tâches de traitement de l'information: de l'information est véhiculée dans l'organisation et manipulée par différentes tâches.

Deux rapports seraient utiles concernant ce traitement de l'information:

- un rapport de diagramme de flux: étant donné une information, un tel rapport décrirait l'environnement et les différentes tâches qui la concernent. Ainsi, en ce qui concerne le bureau, deux types de diagrammes de flux seraient utiles:
  - le flux des informations dans l'organisation et vers l'extérieur: on voudrait savoir par où transite l'information et à qui elle est communiquée.
  - le flux des informations entre les tâches du bureau: quelles sont les tâches qui génèrent telle information, qui la consultent et / ou qui la modifient.
- un deuxième rapport décrirait les données et les résultats des traitements: étant donné une tâche du bureau, on voudrait savoir:
  - quelles sont toutes les informations nécessaires à son exécution ?
  - d'où proviennent ces informations ?
  - quels sont les résultats de son exécution ? Quelles sont les informations modifiées et / ou créées ?
  - à quelles tâches sont destinées ces nouvelles informations ?

Enfin, on pourrait demander un rapport concernant la réquisition des ressources par les différentes tâches: étant donné une tâche, quelles sont les ressources nécessaires à son exécution ? Pendant combien de temps ?

3.2.2 L'INTERROGATION DU SYSTEME D'INFORMATION

Le modèle retient de l'information. A tout moment, on devrait pouvoir l'interroger afin d'obtenir des listes d'éléments répondant à certains critères. Ces critères, plus ou moins complexes, feraient intervenir le nom des objets, leurs propriétés et les relations avec d'autres objets.

Exemples de listes d'éléments en bureautique:

- donner la liste de tous les documents susceptibles d'être envoyés à tel étudiant;
- donner le nom de toutes les personnes responsables de tel projet;
- donner la liste de tous les formulaires manipulés dans le bureau;
- ...

La documentation fournie et l'interrogation du système devraient permettre d'évaluer la qualité du système décrit:

- sa fidélité: le système décrit représente-t-il bien la réalité ?
- sa complétude: tous les aspects de la réalité sont-ils représentés ?
- sa cohérence: n'y a-t-il aucune contradiction ni ambiguité ?

Les rapports documentaires peuvent servir à vérifier la fidélité et la complétude du système. L'interrogation permettrait d'énoncer des critères de cohérence que doit vérifier le système.

3.3 LA SIMULATION

La simulation permettrait d'étudier le comportement d'un système d'information bureautique et d'en évaluer a priori les performances.

Par simulation, on peut voir si une solution conceptuelle est réalisable par rapport aux ressources dont on dispose dans le bureau: les effectifs en personnel, la technologie de bureau existante.

La solution conceptuelle pourrait être évaluée pendant l'étude d'opportunité, en vue de la définition d'un avant-projet de solution; elle pourrait également être évaluée en cours d'analyse conceptuelle détaillée, afin de valider les hypothèses retenues dans l'avant-projet de solution.

4 PRESENTATION DES IDEES MAITRESSES

Dans le chapitre 2, nous ferons une synthèse de la littérature traitant de la conception de systèmes d'information de bureau existants, afin d'en étudier les techniques et méthodes d'analyse.

La construction d'un schéma conceptuel qui assure pleinement sa fonction de communication et de standardisation doit s'appuyer sur des modèles précis. Un modèle est formé de concepts et de règles relatives à leur utilisation. Au chapitre 3, nous proposons différents modèles qui se rapportent aux aspects particuliers du système d'information de bureau à élaborer:

- le modèle de structuration de l'organisation;
- le modèle de structuration des ressources;
- le modèle de structuration de l'information;
- le modèle de structuration des traitements;
- le modèle de la dynamique des traitements.

Dans ce chapitre 3, nous énoncerons également les règles de complétude et de cohérence des modèles proposés.

Au chapitre 4, nous développerons la phase d'implémentation du modèle décrit au chapitre 3. Nous nous limiterons à l'implémentation des objets informationnels (modèle de structuration de l'information). Nous y justifierons le choix d'un système générique pour implémenter notre système détaillé: le système IDA. Nous élaborerons ensuite une solution possible d'implémentation des objets informationnels du bureau.

Dans la conclusion, après un rappel des idées maitresses développées, nous verrons dans quelle mesure les résultats fixés ont été atteints et si le choix du système IDA était réellement un bon choix.

CHAPITRE 2: ETUDE DES MODELES EXISTANTS.

1 INTRODUCTION

L'automatisation du travail de bureau est un des domaines d'application des systèmes d'information qui croit rapidement. Beaucoup d'efforts de recherche ont été réalisés, ces dernières années, dans le développement de nouvelles approches à la conception de systèmes d'information de bureau, considérant des aspects originaux de l'environnement bureautique.

La première étape de l'élaboration du modèle d'information de bureau développé ici consiste à faire une synthèse de la littérature traitant du même sujet, afin d'étudier les techniques et méthodes d'analyse existantes. Donnons un bref aperçu de ces différentes méthodes.

2 OSL: OFFICE SPECIFICATION LANGUAGE

2.1 LE POINT DE DEPART DE OSL

A l'époque où J. Kunin rédige sa thèse [KUN] [HAM], il existe peu de modèles sur lesquels baser l'analyse, la conception et la construction de systèmes d'information de bureau. Les approches conventionnelles de l'automatisation de bureau se basent sur des actions de bas niveau: dactylographier, classer, remplir des formulaires,... . A ce niveau, le travail de bureau est une série d'activités locales et peu connectées; ces activités sont les mêmes de l'employé au chef de département, mais réalisées à des fréquences différentes. Le but du travail est ignoré, ce qui conduit plutôt à mécaniser ce qui existe et non à donner une spécification et une implémentation efficace des systèmes de bureau.

L'idée de J. Kunin est d'avoir une vue plus abstraite du bureau: il se place au niveau fonctionnel. Les fonctions du bureau sont des ensembles de procédures de bureau structurées ou non; les procédures sont des constructions de haut niveau qui organisent et ordonnent les activités individuelles du bureau.

2.2 OSL: UN LANGAGE STRUCTURE

Le but principal de la conception de OSL était de développer un langage structuré, de façon à aider un analyste dans la construction de spécifications. Pour construire un tel langage, J.Kunin se base sur 4 critères:

- Le langage doit être formel , c'est-à-dire avoir une syntaxe et une sémantique bien déterminées. Ce critère élimine les ambiguités et l'imprécision des spécifications en langage naturel.

- Le langage doit être orienté vers le problème , c'est-à-dire que les primitives du langage correspondent aux structures et aux activités du bureau.

- Le langage doit être modifiable , afin d'éviter que le système ne devienne obsolète et inutilisable.

- Le langage doit posséder une structure hiérarchique , car le même niveau de détail n'est pas approprié aux différents usages.

2.3 LES PRINCIPES DE BASE DE OSL

Il y a 4 principes sous-jacents à OSL:

- 1er principe: Il existe une structure dans le bureau. Le bureau est un système et, en tant qu'unité organisationnelle, il a une mission. Les composants du système incluent:
  - les gens: les activités des travailleurs sont conçues pour réaliser la mission du bureau.
  - l'équipement, l'information, l'espace: ils aident les gens à réaliser la mission du bureau.

- 2ème principe: Il y a un haut degré de similitude de la structure et de l'activité parmi les procédures de différents bureaux. Si l'on identifie ces structures et activités (par des études de cas et des analyses de travaux apparentés), on peut les réunir dans un langage formel qui offrira à l'utilisateur les caractéristiques du bureau.

- 3ème principe: Les procédures de bureau sont fondamentalement simples. Cependant, des détails d'implémentation et un traitement d'exception desorganisé les rendent souvent complexes. Le but de OSL est d'éviter cette "complexité" afin de découvrir la structure de base des procédures de bureau, généralement facile à comprendre et à décrire.

- 4ème principe: Les objets de papier ne sont pas la base fondamentale des procédures de bureau. Ils permettent seulement de collecter et de transmettre de l'information au sujet d'objets plus abstraits.

2.4 SYNTHESE

La structure et les caractéristiques de OSL dérivent des principes énoncés ci-dessus. Une spécification OSL est une description, dans la notation OSL, de la structure et des opérations d'une partie d'une organisation. Cette spécification possède deux composants: l'environnement et la spécification opérationnelle.

L'environnement est une description de la structure statique du bureau. Il décrit les objets traités, le contexte organisationnel et l'information utilisée. Il établit le vocabulaire nécessaire à la partie opérationnelle de la spécification. La description de l'environnement de bureau s'exprime en termes d'entités et de leurs attributs, de relations inter-entités et de collections d'entités ou classes.

Cette spécification d'environnement comprend:

- une description organisationnelle interne du bureau: elle identifie l'organisation, les hiérarchies de personnel, les responsabilités, les communications,... .
- une description du contexte externe à l'organisation: elle identifie les aspects extérieurs utiles pour le bureau décrit.

La spécification opérationnelle inclut différents niveaux d'abstraction. Au plus haut niveau, une fonction représente la gestion d'un ensemble d'entités, les "ressources" de la fonction, au cours du temps. Elle fournit un résumé des procédures et des événements utiles à la gestion de cette ressource. Une procédure est un ensemble d'activités, d'états et d'événements externes ou internes qui déclenchent les activités si les états sont vérifiés par les objets manipulés.

2.5 OAM ET OSL: UN OUTIL COMPLET

Parallèlement à OSL, on a développé une méthodologie d'analyse de bureau: OAM. Celle-ci aide l'analyste à comprendre la structure du bureau qu'il étudie et à organiser des interviews et la rédaction de rapports.

2.6 CONCLUSION

OAM et OSL offrent un outil complet, l'un aidant à la collecte des informations, l'autre permettant la modélisation de ces informations.

3 OFFIS: OFFICE INFORMATION SPECIFIER

3.1 PRESENTATION GENERALE

Le système OFFIS [BRA] a été conçu pour faciliter l'analyse interactive et itérative, ainsi que le processus de conception, de systèmes bureautiques. Pour ce, il offre au concepteur de bureaux automatisés une méthode flexible de documentation et d'analyse des caractéristiques et des contraintes d'un système.

Le système OFFIS a trois fonctions de base:

- Documentation : OFFIS offre un modèle descriptif pour documenter les procédures et les activités d'un bureau.
- Analyse : OFFIS peut être utilisé comme outil d'analyse afin d'évaluer la complétude et la cohérence des procédures documentées.
- Conception : OFFIS fournit un support d'aide à la conception et au développement de systèmes d'information, au moyen d'outils de simulation et de conception.

3.2 LE LANGAGE OFFIS

Le langage OFFIS a pour but d'offrir une méthode appropriée pour énoncer l'information requise lors du développement d'un système de bureau automatisé.

Le langage OFFIS comprend des objets et des relations permettant la spécification des hiérarchies dans les bureaux, la description des traitements, la plannification d'événements et les différentes formes de communications internes et externes.

3.3 L'ANALYSEUR OFFIS

L'analyseur OFFIS examine les définitions des besoins du bureau exprimées par l'utilisateur dans le langage OFFIS. Il analyse la syntaxe et certaines relations sémantiques et, itérativement, il construit une base de données qu'il utilisera ensuite pour la génération de rapports.

3.4 LES RAPPORTS FOURNIS PAR L'ANALYSEUR OFFIS

Divers rapports d'analyse et d'évaluation sont disponibles. Ils fournissent une vue générale de la conception du système. Ils décrivent:

- la hiérarchie organisationnelle;
- les fonctions de l'organisation;
- les processus associés à chaque personne;
- les flux de données;
- les incomplétudes et les incohérences dans la conception.

3.5 CONCLUSION

L'utilisation du système OFFIS permet une description dynamique du bureau et parallèlement, donne une documentation de cette évolution.

Cependant, les opérations et le traitement de celles-ci sont simplifiés, ce qui limite l'utilité de l'approche.

4 OBE: OFFICE-BY-EXAMPLE

4.1 INTRODUCTION

Le système OBE [ZLO] traduit la volonté de combiner les aspects du traitement de texte, du traitement des données, de la création de rapports, du graphisme et du courrier électronique. Il offre à l'utilisateur différents utilitaires et un langage lui permettant de spécifier ses propres applications.

4.2 LE LANGAGE OBE

Le langage OBE est un langage bidimensionnel, non procédural. Les composants de ce langage sont:

- l' input : On peut acquérir les données du système OBE
  - de façon interactive;
  - en provenance d'autres utilisateurs du système;
  - à partir de différentes bases de données.

- les structures de données : Elles sont bidimensionnelles et sont définies par l'utilisateur final. Ces structures comprennent:
  - des objets-données: RELATIONS, FORMES, RAPPORTS, MENUS, ... Ces objets-données sont des tables qui comprennent des éléments variables.
  - des objets-programme, qui sont une collection d'objets-données associés à leurs opérations.

- les types de données : Les types existant dans OBE sont CHAR, FIXED, FLOAT, DATE, TIME.

- les conditions : On peut poser des conditions sur les valeurs que peuvent prendre les éléments variables d'un objet.

- les requêtes de base de données: OBE permet à l'utilisateur de définir et d'interroger une ou plusieurs bases de données. Le langage de requêtes offre toutes les opérations algébriques relationnelles: sélection, projection, jointure, intersection, différence.

- les branchements. Les branchements vers différentes actions peuvent être réalisés au moyen de déclencheurs. Quand un déclencheur est activé, il en résulte l'exécution d'une ou plusieurs actions.

- la construction de menus : OBE permet à l'utilisateur de créer ses propres menus (contrairement aux systèmes traditionnels qui, le plus souvent offrent des menus préspécifiés avec des objets et des fonctions préprogrammés).

- les commandes : OBE utilise différentes commandes pour exécuter, envoyer, imprimer, détruire ou mettre à jour.

- l' output : OBE permet d'afficher à l'écran, d'imprimer ou d'envoyer sur disque des données d'output.

4.3 LES COMPOSANTS DU SYSTEME OBE

Le système OBE se compose de:

- un éditeur d'écran qui permet de définir les données au moyen d'une gestion de fenêtre, et dont les fonctions couvrent également une partie du traitement de texte.

- des fonctions de formatage des données offrant différentes options de présentation.

- des processeurs de définition de base de données où l'utilisateur pourra stocker de l'information et qu'il pourra ensuite consulter.

- un courrier électronique permettant à plusieurs utilisateurs de s'échanger de l'information.

- un gestionnaire des déclencheurs qui permet l'automatisation des procédures de bureau.

- des utilitaires pour créer, stocker et retrouver des documents.

- le moyen de définir des menus et de stocker des programmes, et donc de développer des applications plus ou moins complexes.

4.4 CONCLUSION

Le système OBE combine le traitement de texte, le traitement des données et la communication. Il permet à l'utilisateur de créer ses propres menus pour faire appel à des programmes ou même, d'automatiser des procédures au moyen des déclencheurs.

On peut remarquer cependant que le système OBE met l'accent plus sur la recherche d'un interface avec l'utilisateur, agréable et facile à utiliser plutôt que sur l'analyse et la spécification des systèmes de bureau.

5 OFS: OFFICE FORM SYSTEM

5.1 INTRODUCTION

Tsichritzis [TSI] propose une approche des systèmes d'information de bureau basée sur les formes(a). Cette approche met l'accent sur un mode structuré de communication et de stockage d'information.

5.2 LES FORMES

Tsichritzis introduit les formes en tant qu'abstraction et généralisation des formes de papier. Des opérations standards sont définies sur les formes, mais cet ensemble peut être étendu à des opérations spécifiques à des types de formes, ce qui nécessite la définition de contraintes d'integrité et d'effets de bord.

Définitions des formes.

Un type de forme représente un type de données défini pour les formes. Il consiste en:

- un ensemble d'attributs X(0), X(1), ..., X(n), tels que X(0) est un identifiant;
- un ensemble de procédures d'opérations P(1), P(2), ..., P(n) que l'on spécifie lorsqu'on veut faire appel à une opération non standard.

Une occurrence de forme est une occurrence du type. C'est un ensemble de valeurs x(0), x(1), ..., x(n) correspondant aux attributs X(0), X(1), X(2), ..., X(n) et un ensemble de valeurs obtenues par application des procédures d'opérations.

Un patron (ce mot ayant le même sens que dans le langage des couturières) de forme est une correspondance entre une occurrence de forme et une représentation via un moyen de communication particulier.

Les messages, documents et formulaires peuvent être représentés au moyen du concept de forme.

(a) Je traduis ainsi le mot anglais "forms" afin de ne pas confondre le concept de "forme" au sens de Tsichritzis et le concept de formulaire habituellement utilisé dans les bureaux.

Les opérations sur les formes.

Les opérations s'appliquent à des occurrences de formes. Tsichritzis met en évidence différentes opérations:

- entrer les valeurs d'attribut d'une forme, en conservant ou non les anciennes valeurs, interactivement ou à partir d'un fichier;
- modifier des valeurs d'attribut;
- stocker une forme;
- copier une forme;
- détruire une forme, avec possibilité de l'archiver;
- communiquer des formes entre stations dans le système;
- accéder à une ou plusieurs formes séquentiellement, selon une valeur d'attribut ou par requête.

A chacune de ces opérations, on peut associer des procédures qui vont permettre d'en contrôler les effets de bord.

5.3 LES PROCEDURES LIEES AUX FORMES

Les opérations sur les formes sont initialisées directement par l'utilisateur. Cependant, il existe des situations où l'on voudrait spécifier des actions que le système exécuterait automatiquement. Ces procédures automatiques sont déclenchées lorsque certaines conditions sont vérifiées; elles exécutent certaines actions puis testent des postconditions. Si les postconditions ne sont pas vérifiées, les procédures se terminent par un ensemble d'opérations "d'échec".

5.4 LES FLUX DE FORMES

Tsichritzis offre des techniques pour examiner les flux dans le bureau. Ces techniques permettent:

- d'analyser les opérations exécutées sur les formes afin d'éviter les effets de bord non souhaités si le système est mal conçu.

- d'étudier la structure du bureau et les chemins suivis par les formes afin de mettre en évidence les erreurs de routage et les goulots d'étranglement.

6 LE MODELE DE DONNEES DE GIBBS ET TSICHRTZIS

Tsichritzis, en collaboration avec Gibbs [GIB], a aussi présenté un modèle de données conçu spécifiquement pour l'environnement de bureau.

Le modèle est orienté-objet et fait appel aux mécanismes d'abstraction (généralisation - spécialisation - agrégation) utilisés dans la modélisation conceptuelle et la représentation de connaissance.

La structure de l'objet est spécifiée déclarativement et consiste en une hiérarchie de propriétés et des associations de composition. Des contraintes sont exprimées de façon procédurale en utilisant des définitions de domaines (qui définissent les formats des valeurs de propriétés) et des déclencheurs (qui sont utilisés pour tester des préconditions et des postconditions). Les types de données permis comprennent l'audio, l'image et le texte en plus des types que l'on trouve traditionnellement dans les langages de programmation. On peut définir des patrons (de nouveau, ce mot est pris au sens des couturières) pour présenter les objets via différents media.

Ce modèle peut être utilisé pour représenter le bureau:

- Les définitions d'objets, les contraintes et les patrons permettent de décrire les objets et l'environnement du bureau (message, formulaire, document, fichier, dossier, personne, ...) et les relations entre ces objets.
- Les déclencheurs permettent de décrire des procédures de bureau. La longueur et la complexité de ces procedures sont cependant limitées.

7 LE SYSTEME POISE

7.1 INTRODUCTION

Le système POISE [CRO] fournit un support de tâches de bureau sur base de hiérarchies de description de tâches (ou procédures). Les descriptions de procédures spécifient les pas typiques à la tâche, les appels d'outils qui correspondent à ces pas et leur objectif. Le système POISE peut être utilisé pour automatiser des tâches de routine ou fournir de l'aide dans des situations plus complexes.

7.2 POISE: UN INTERFACE INTELLIGENT

POISE agit comme un interface intelligent entre l'utilisateur et les outils disponibles dans un système de bureau. Trois types d'information sont utilisés par cet interface:

- la librairie des procédures: elle contient les descriptions des procédures.
- la base de données sémantique: elle contient les descriptions des objets utilisés dans les procédures et les descriptions des outils disponibles.
- le modèle d'état d'un utilisateur particulier: il inclut des instanciations partielles des descriptions de procédures avec des paramètres dérivés d'actions spécifiques, ainsi que des instanciations d'objets de la base de données sémantique.

7.3 LES POSSIBILITES DU SYSTEME POISE

Le système POISE permet:

- l'automatisation des tâches de routine;
- la proposition d'actions alternatives lorsque l'on se trouve à un point de décision;
- la reconnaissance d'actions inappropriées dans le contexte de ce que l'utilisateur est en train de faire;
- l'interrogation au moyen de requêtes en langage naturel et la génération des descriptions de l'état courant.

7.4 DEFINITION DES PROCEDURES

Pour représenter les séquences possibles d'actions concurrentes dans une procédure, POISE utilise une version modifiée d'un langage de description d'événements (EDL).

La syntaxe algorithmique de la procédure est spécifiée par une clause IS et ses paramètres par une clause WITH. On peut imposer des contraintes sur les valeurs et les relations des attributs des procédures; elles sont spécifiées par la clause COND.

Une clause PRECONDITION spécifie les conditions requises pour qu'une procédure commence et une clause SATISFACTION contient les objectifs satisfaits par une procédure.

Différents opérateurs (séquence, alternative, parallélisme, option, itération) spécifient la séquence des procédures et des actions primitives constituant une tâche.

Les procédures sont définies de façon hiérarchique et le niveau inférieur correspond approximativement à des appels d'outils. La correspondance réelle entre procédures "primitives" et outils est réalisée par une table.

Ce formalisme est adéquat pour un analyste de système, mais non acceptable pour une présentation à des utilisateurs. La recherche se poursuit pour développer un interface agréable.

7.5 LA BASE DE DONNEES SEMANTIQUE

La base de données sémantique décrit les objets du bureau et de son environnement. Les objets peuvent avoir des attributs simples ou composés; ils peuvent aussi être attributs d'autres objets. Le modèle spécifie des contraintes sur les objets et les attributs.

7.6 CONCLUSION

Le système POISE permet d'automatiser des actions qui correspondent directement à des outils de bureau et de fournir de l'aide dans la prise de décision. Des recherches se poursuivent, notamment en ce qui concerne l'interface et la représentation de la connaissance.

8 CONCLUSION GENERALE

L'étude de différents modèles permet de tirer des conclusions quant à leurs caractéristiques communes et à leurs différences.

8.1 LES CARACTERISTIQUES COMMUNES DES SYSTEMES D'INFORMATION DE BUREAU

Ces caractéristiques permettent de comparer la conception d'un système d'information de bureau et celle d'un système d'information traditionnel.

- les données de bureau : On introduit des données non structurées contenues dans des messages, textes, graphiques et communications orales.

- les activités de bureau : Certaines activités sont non structurées. Il n'y a pas seulement des activités qui peuvent se traduire de façon algorithmique, mais certaines activités font appel à la décision.

- les interconnexions des éléments : En général, les éléments nécessaires pour exécuter le travail de bureau sont distribués parmi plusieurs travailleurs, dans le même département ou dans des départements différents. La communication entre travailleurs et avec le monde extérieur est une des principales fonctions du bureau.

- l'évolution du bureau : Cette évolution peut être relative:
  - aux contraintes: exemple: changement de responsabilité par rapport à une tâche.
  - aux activités: exemple: modification des activités dans le temps et selon l'évolution de la technologie.
  - aux objets: exemple: de nouveaux requis légaux entrainent un changement des documents.

- les caractéristiques d'usage : Les systèmes d'information de bureau sont fortement interactifs. Les utilisateurs peuvent être non professionnels et très diversifiés, d'où le besoin d'un système agréable et facile à utiliser.

- l'intégration des fonctions : De nombreuses fonctions, telles que la communication, la manipulation et la recherche d'information, la gestion des tâches, sont liées et utilisées par le même travailleur dans une séquence rapide. Il s'agit d'étudier le bureau comme un tout et non comme une collection de tâches et d'équipement isolés. Cette intégration permet de réduire la complexité de l'interface avec l'utilisateur, de contrôler le flux d'information et d'accroitre l'efficacité du bureau.

- l'impact de la technologie : Les technologies des systèmes bureautiques évoluent rapidement. Dès lors, les méthologies de bureau devraient être aussi indépendantes que possible des détails d'implémentation.

8.2 LES DIFFERENTES ETAPES DE LA CONCEPTION D'UN SYSTEME BUREAUTIQUE

On peut distinguer 3 phases dans la conception d'un système d'information de bureau:

- l'analyse des besoins : On étudie le bureau et les fonctions exécutées.

- la spécification des besoins : On utilise un modèle conceptuel de bureau. Il faut une description formelle et aussi complète que possible de tous les aspects du travail de bureau. Cette description doit être compréhensible non seulement par le concepteur mais aussi par l'utilisateur.

- l'implémentation du système.

La littérature analyse plus particulièrement le problème de la spécification formelle du bureau plutôt que de proposer une approche méthodologique complète.

8.3 CLASSIFICATION

Les modèles peuvent être classés sur base des éléments fondamentaux qu'ils prennent en considération:

- les modèles basés sur les donnés : Les types de données et les opérations sur les données (stockage, recherche, manipulation, transmission) sont les éléments de base. Les activités de bureau sont alors vues comme une série d'opérations sur les données et sont prises en considération individuellement.
  exemples: OBE, OFFIS, le modèle de données de Gibbs et Tsichritzis.

- les modèles basés sur les processus : Ils analysent et décrivent le travail de bureau en considérant les différentes activités exécutées en concurrence par les utilisateurs et le système. On présente les activités de façon coordonnée. L'approche est fondée sur une vision intégrée de toutes les activités exécutées dans le bureau.
  exemple: OSL.

- les modèles mixtes : Ils supposent plus d'un type d'élément comme base pour la spécification du système.
  exemple:OFS.

CHAPITRE 3: MIB: UN MODELE D'INFORMATION DE BUREAU.

1 INTRODUCTION

Dans ce chapitre, nous développerons les deux premières phases de la conception d'un système d'information: l'analyse et la spécification des besoins. Nous allons élaborer ici un modèle d'information de bureau qui prendra en compte les caractéristiques originales du bureau: le MIB.

2 PREMIERE ETAPE: L'ANALYSE DES BESOINS

La première étape de la conception d'un modèle d'information de bureau consiste en une étude approfondie du bureau. Cette étude fut réalisée au secrétariat administratif de l'institut d'informatique. Il s'agissait d'examiner le flux administratif lié aux étudiants venant s'inscrire à l'institut.

Comment avons-nous procédé pour cette étude ?

Nous nous sommes basés sur la méthode d'analyse OAM mentionnée dans l'introduction [**HAM**].

Dans un premier temps, une entrevue avec la secrétaire a permis de rassembler toutes les données nécessaires à la description du bureau. Pour le flux considéré, nous avons posé les questions suivantes:

- quelles sont les différentes étapes relatives à l'inscription des étudiants ?

et pour chaque étape:

- QUAND réalise-t-on cette étape ?
- QUE fait-on ?
- COMMENT le fait-on ?

Après l'entrevue, nous avons rédigé un brouillon afin de tirer des informations collectées une image claire des opérations de bureau, en faisant apparaitre les différentes étapes de traitement, les objets manipulés et les ressources utilisées. Nous avons également établi une liste des choses qui nous paraissaient incomplètes ou incorrectes. Une seconde entrevue nous a permis de combler ces lacunes.

Disposant de la description complète du bureau, nous avons alors structuré cette information pour obtenir le rapport final.

Le résultat de cette analyse se trouve à l'annexe 1.

3 DEUXIEME ETAPE: LA SPECIFICATION FORMELLE

Comme nous l'avons vu dans l'étude des modèles existants, cette étape consiste à donner une description formelle et aussi complète que possible du travail de bureau.

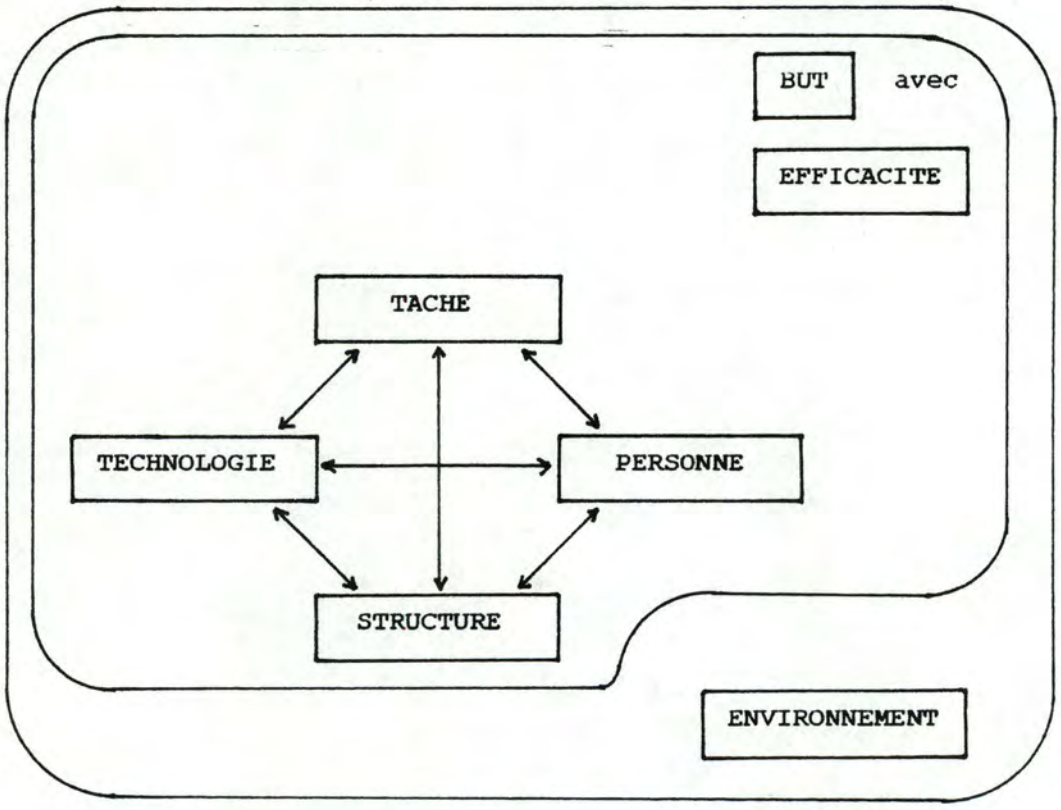
3.1 LA STRUCTURE DU MODELE

L'analyse des composants du bureau réalisée dans la première étape nous a permis de cerner les différents concepts manipulés. Comment structurer ces concepts isolés de façon à faire apparaitre les liens existant entre chacun d'eux et ainsi obtenir une formalisation cohérente ?

Notre idée fut de se baser sur le diamant de Leavitt, qui offre une synthèse des éléments constitutifs d'une organisation et des relations entre ceux-ci.

3.1.1 LE DIAMANT DE LEAVITT

Rappelons brièvement comment se présente le diamant de Leavitt. On pourra trouver plus de détails dans [LES].



Le diamant de Leavitt

Le diamant se compose de quatre facettes principales: les personnes, les tâches, la technologie et la structure.

- L'organisation est une entité sociale, un ensemble de personnes réunies pour poursuivre pendant un certain temps un même but.

- Afin de réaliser ce but, les personnes exécutent différentes tâches en utilisant de la technologie. Chacun joue un rôle spécifique.

- Personnes, tâches et technologie sont intégrées dans une structure qui détermine les relations entre les tâches, les responsabilités et le pouvoir des personnes et ainsi, permet à l'organisation de fonctionner.

La réalisation du but doit répondre à certains critères d'efficacité quantitative ou qualitative.

L'organisation dépend de l' environnement avec lequel elle est en relation.

3.1.2 APPLICATION AU BUREAU

Le bureau est une organisation réduite; on peut donc le décrire au moyen du diamant de Leavitt.

a. Les tâches : Dans le cadre de la bureautique, on se limite aux tâches traitant de l'information. Il existe différents types de tâches qui sont fonction de la stabilité ou du caractère variable et de la complexité de la tâche.

b. La technologie : Son introduction dans les bureaux est récente.

c. Les personnes : Elles interviennent dans le bureau en tant que:
   - personnes physiques jouant un rôle dans l'organigramme de l'organisation;
   - facteur de production pour l'exécution des tâches.

   La technologie et les personnes constituent les ressources du bureau.

d. La structure : On se limitera à une structure hiérarchique. Mais le modèle pourrait être étendu à d'autres styles de gouvernement.

3.2 LA FORMALISATION DES CONCEPTS

3.2.1 INTRODUCTION

Le diamant de Leavitt souligne les aspects particuliers du modèle général d'information de bureau (MIB). Correspondant à chacun de ces aspects, nous définirons différents modèles particuliers. Ces modèles sont parallèles à ceux mis en évidence par F. Bodart dans [BOD].

- Le modèle de structuration de l'organisation : Il permet de définir la facette "structure" du diamant de Leavitt: l'organigramme de l'organisation, les responsabilités et le rôle des différentes personnes dans la réalisation d'un but fixé.

- Le modèle de structuration de l'information : Il sert à définir l'information traitée dans le bureau.

- Le modèle des ressources : Il décrit les facettes "technologie" et "personnes" du diamant de Leavitt.

- Le modèle de structuration des traitements : Il permet une décomposition des traitements en différentes tâches et une découpe élémentaire des tâches en opérations primitives. Il décrit donc la facette "tâches" du diamant de Leavitt.

- Le modèle de la dynamique des traitements : Il permet de décrire les enchainements entre tâches et, pour chaque tâche, entre opérations primitives.

3.2.2 LE MODELE ENTITE - ASSOCIATION

Le modèle entité - association est un modèle conceptuel général - non spécifique à un problème particulier - que l'on utilisera pour décrire les concepts organisationnels et informationnels et les ressources du MIB.

Nous allons rappeler les concepts de base de ce modèle. On peut en trouver un exposé plus détaillé dans [BOD].

3.2.2.1 DEFINITIONS DES CONCEPTS DE BASE

ENTITE:
Une entité est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité peut posséder des attributs.

ASSOCIATION:
Une association est définie par une correspondance entre deux ou plusieurs entités où chacune assume un rôle donné. Une association peut posséder un ou plusieurs attributs.

ATTRIBUT:
C'est une caractéristique ou qualité d'une entité ou d'une association. Elle peut prendre une ou plusieurs valeurs ou groupes de valeurs.

TYPE D'ENTITE (D'ASSOCIATION / DE VALEUR D'ATTRIBUT):
C'est la classe de toutes les entités (associations / valeurs) possibles du réel perçu qui vérifient la définition constitutive du type.

PROPRIETE D'UN TYPE D'ASSOCIATION:
CONNECTIVITE
Soit R (E[1], E[2], ... , E[i], ... , E[n]) un type d'association défini sur les types d'entités E[1], E[2], ..., E[i], ..., E[n]. La connectivité de R est définie par un ensemble de couples d'entiers (min[i], max[i]), où:

- min[i] indique le nombre minimum d'occurrences de R auquel toute occurrence de E[i] doit participer à tout moment;
- max[i] indique le nombre d'occurrences possibles de R pour toute occurrence de E[i].

Les valeurs les plus fréquemment utilisées sont:

- min[i] = 0 ou 1;
- max[i] = 1 ou N (infini).

PROPRIETES DES ATTRIBUTS:
ATTRIBUT SIMPLE OU REPETITF
Un attribut est simple si pour une occurrence d'un type d'entité ou d'association, il ne peut prendre qu'une seule valeur.
Il est répétitif si pour une occurrence d'un type d'entité ou d'association, il peut prendre plusieurs valeurs d'un même type.

ATTRIBUT ELEMENTAIRE OU DECOMPOSABLE
Un attribut est décomposable si à une occurrence d'un type d'entité ou d'association, il fait correspondre un groupe de valeurs de types différents et peut être décomposé, au plus, en autant d'attributs qu'il y a de types différents dans le groupe de valeurs.
Un attribut non décomposable est élémentaire.

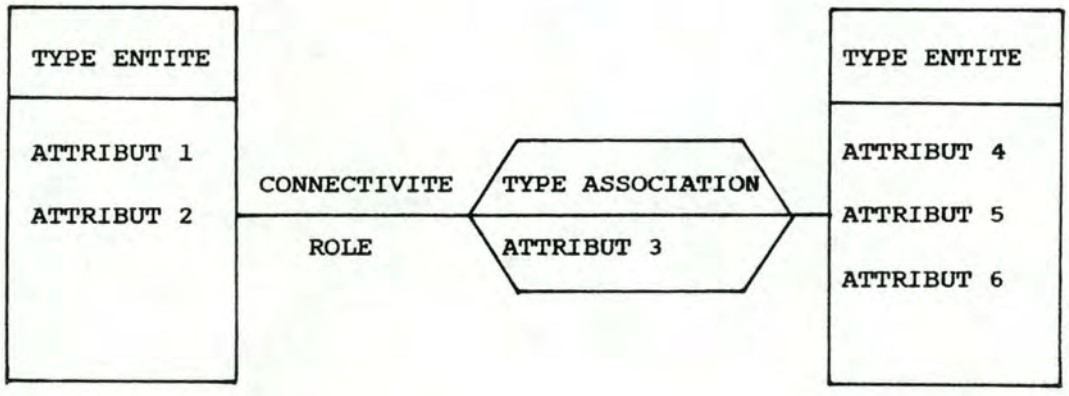
ATTRIBUT OBLIGATOIRE OU FACULTATIF
Un attribut obligatoire doit prendre une valeur effective pour chaque occurrence du type qu'il caractérise.
Un attribut facultatif peut ne pas prendre de valeur pour certaines occurrences du type qu'il caractérise; cet attribut n'a pas de signification pour ces occurrences.

LES CONTRAINTES D'INTEGRITE:
Les contraintes d'intégrité sont des propriétés non représentées par les concepts de base du modèle et que les informations du système doivent vérifier. Les contraintes d'intégrité sont statiques ou dynamiques et portent

- sur les types d'entités: contraintes d'existence, identifiant;
- sur des associations: contraintes d'existence, d'exclusion, d'inclusion, identifiant;
- sur des attributs: contraintes de valeur, dépendance fonctionnelle.

3.2.2.2 REPRESENTATION GRAPHIQUE

On peut représenter graphiquement les concepts de base comme suit:



3.2.2.3 DESCRIPTION DU MIB

Nous décrirons les différents concepts du MIB par des entités et les relations entre ces concepts par des associations.

Pour les objets, nous donnerons:

- une définition générale décrivant l'objet
- les attributs de l'objet avec
  - une description
  - leurs propriétés
  - le type de valeur qui les caractérise
- l'identifiant de l'objet

Pour les associations, nous donnerons:

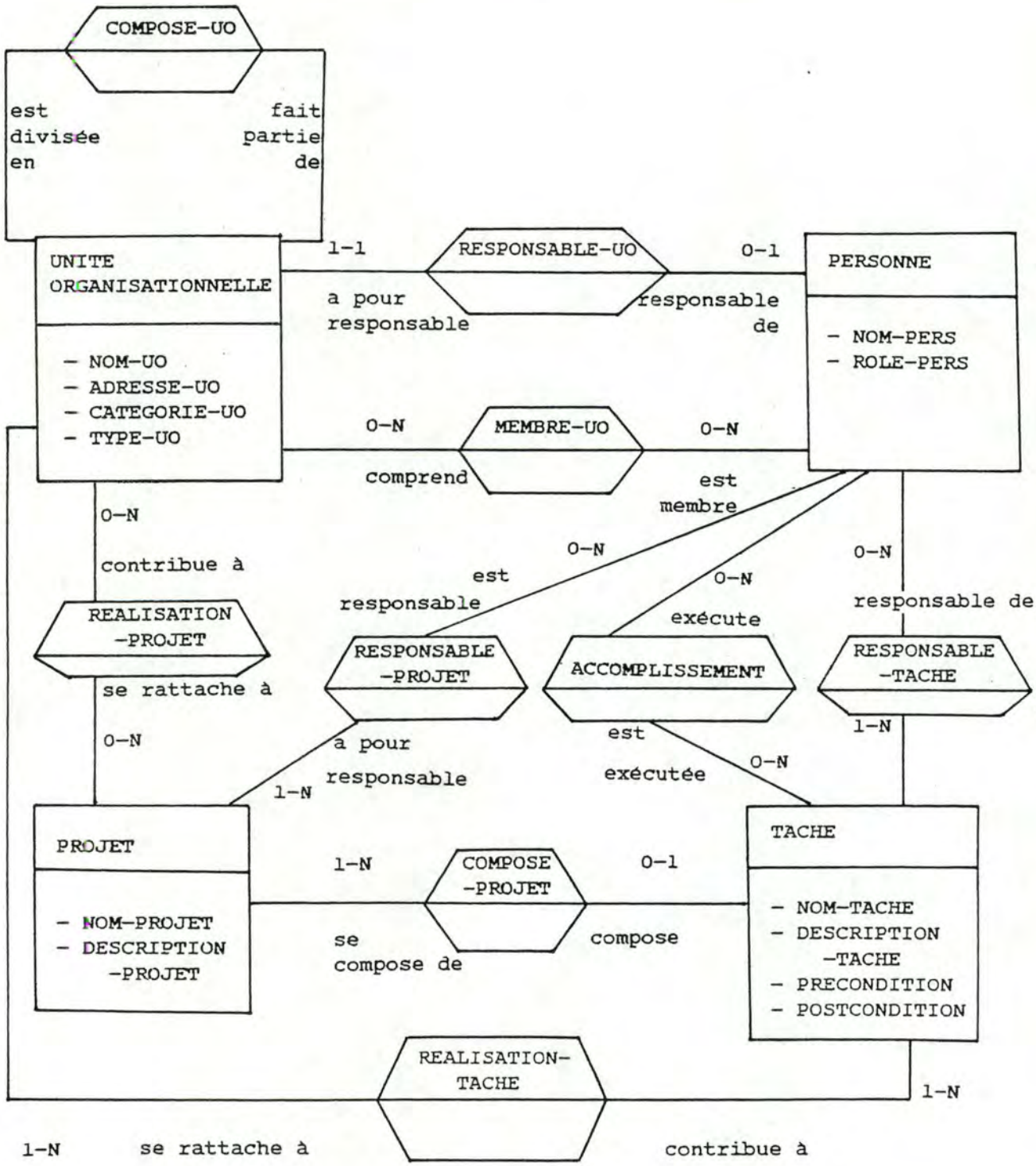
- une description de la relation représentée
- les entités sur lesquelles l'association est définie
- le rôle que chacune de ces entités assume dans l'association
- la propriété de connectivité de l'association.

Nous définirons en outre les contraintes d'intégrité nécessaires.

3.2.3 LE MODELE DE STRUCTURATION DE L'ORGANISATION

Ce modèle permet de décrire l'organisation, sa structure, les responsabilités des différentes personnes ou groupes de personnes. Le but de l'organisation peut être atteint par la réalisation des différentes taches décrites.

3.2.3.1 REPRESENTATION GRAPHIQUE



3.2.3.2 DESCRIPTION DES ENTITES

Entité UNITE ORGANISATIONNELLE.

a. Définition.

Une organisation (qui est l'unité organisationnelle la plus importante) est un groupe permanent de personnes qui ont un objectif spécifique qu'elles cherchent à réaliser.

L'organisation est divisée en unités organisationnelles.

Chaque unité organisationnelle a un responsable ainsi que des membres.

Une unité organisationnelle peut:
- soit être décomposée en d'autres unités organisationnelles
- soit être constituée de personnes qui en sont les membres.

On distinguera une unité organisationnelle particulière: l'environnement de l'organisation.

b. Attributs.

- NOM-UO.
  Définition:
  C'est le nom de l'unité organisationnelle.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ADRESSE-UO.
  Définition:
  C'est l'adresse de l'unité organisationnelle.
  Propriétés:
  C'est un attribut simple, décomposable et obligatoire.
  Format:
  Cet attribut peut être décomposé en :
  - NR-RUE-UO: numéro de localisation de l'unité organisationnelle. C'est un entier.
  - RUE-UO: nom de la rue. C'est une suite finie de caractères.
  - CODE-POSTAL-UO: code postal de la localité de l'unité organisationnelle. C'est une suite de 4 chiffres.

- CATEGORIE-UO.
  Définition:
  Cet attribut dit à quel titre l'unité organisationnelle est une partie de l'organisation. On ne retiendra pas de valeurs standards pour ces catégories sinon que la catégorie la plus importante est l'ORGANISATION. Les valeurs pourront être choisies par l'utilisateur, mais les unités organisationnelles d'un même degré de décomposition devront être de la même catégorie.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- TYPE-UO.
  Définition:
  Cet attribut donne le type d'activité de l'unité organisationnelle. Cette activité peut être décisionnelle, opérationnelle, ou mixte.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères qui peut prendre les valeurs: DECISIONNELLE, OPERATIONNELLE OU MIXTE.

c. Identifiant.

Une unité organisationnelle est identifiée par son nom NOM-UO.

Entité PERSONNE.

a. Définition.

C'est toute personne physique.

Une personne possède un nom.

Chaque personne joue un rôle dans l'organisation. En bureautique, on peut distinguer quatre types d'acteurs en fonction des rôles: administrateur, professionnel, clerc, secrétaire.

b. Attributs.

- NOM_PERS.
Définition:
C'est le nom de la personne.
Propriétés:
C'est un attribut simple, élémentaire et obligatoire.
Format:
C'est une suite finie de caractères.

- ROLE-PERS.
Définition:
C'est le rôle que la personne joue dans l'organisation.
Propriétés:
C'est un attribut simple, élémentaire et obligatoire.
Format:
C'est une suite finie de caractères qui peut prendre les valeurs: ADMINISTRATEUR, PROFESSIONNEL, CLERC, SECRETAIRE.

c. Identifiant.

Une personne est identifiée par son nom NOM-PERS et le nom NOM-UO de l'organisation dont elle fait partie.

Entité TACHE.

a. Définition.

Une tâche remplit un objectif, une fonction. C'est une suite d'actions à exécuter dans un ordre convenu (cf dynamique) pour résoudre un problème.

b. Attributs.

- NOM-TACHE.
  Défnition:
  C'est un identifiant qui devrait indiquer la nature du traitement.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- DESCRIPTION-TACHE.
  Définition:
  Texte concis, en français, décrivant l'objectif de la tâche.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un texte.

- PRE-TACHE.
  Définition:
  Condition qui doit expliciter les propriétés des arguments qui doivent être vérifiées avant toute exécution de la tâche pour que celle-ci s'exécute correctement. Ces propriétés porteront sur la valeur de l'ETAT (voir la définition des objets informationnels) des objets donnés en arguments et éventuellement, sur la valeur de certains autres attributs.

- POST-TACHE.
  Définition:
  Condition qui doit expliciter les propriétés des résultats de la tâche, qui doivent être satisfaites à la fin de toute exécution de la tâche si celle-ci s'est exécutée correctement.

c. Identifiant.

Une tâche est identifiée par son nom NOM-TACHE.

Entité PROJET.

a. Définition.

Un projet est un ensemble de tâches temporaires réalisées par des délégués d'une ou plusieurs unités organisationnelles.

b. Attributs.

- NOM-PROJET.
  Définition:
  C'est le nom du projet.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- DESCRIPTION-PROJET.
  Définition:
  Description concise du projet.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un texte.

c. Identifiant.

Un projet est identifié par son nom NOM-PROJET.

3.2.3.3 DESCRIPTION DES ASSOCIATIONS

Association COMPOSE-UO.

Description	Une unité organisationnelle peut se décomposer en plusieurs autres unités organisationnelles. Cette association permet de décrire la structure hiérarchique de l'organisation.
Liaison entre	UNITE ORGANISATIONNELLE (1) et UNITE ORGANISATIONNELLE (2).
Rôle	"est divisée en" pour UNITE ORGANISATIONNELLE (1) "fait partie de" pour UNITE ORGANISATIONNELLE (2).
Connectivité	0-N pour UNITE ORGANISATIONNELLE (1) 0-1 POUR UNITE ORGANISATIONNELLE (2).

Association RESPONSABLE-UO.

Description	Une personne peut être responsable d'une unité organisationnelle.
Liaison entre	UNITE ORGANISATIONNELLE et PERSONNE.
Rôle	"est responsable de" pour PERSONNE "a pour responsable" pour UNITE ORGANISATIONNELLE.
Connectivité	1-1 pour UNITE ORGANISATIONNELLE 0-1 pour PERSONNE.

Association MEMBRE-UO.

Description	Au dernier niveau hiérarchique, une unité organisationnelle peut être composée de personnes qui en sont les membres.
Liaison entre	UNITE ORGANISATIONNELLE et PERSONNE.
Rôle	"comprend" pour UNITE ORGANISATIONNELLE "est membre de" pour PERSONNE
Connectivité	0-N pour UNITE ORGANISATIONNELLE 0-N pour PERSONNE.

Association REALISATION-TACHE.

Description	Une tâche se rattache à une ou plusieurs unités organisationnelles.
Liaison entre	UNITE ORGANISATIONNELLE et TACHE.
Rôle	"contribue à" pour UNITE ORGANSATIONNELLE "se rattache à" pour TACHE.
Connectivité	1-N pour UNITE ORGANISATIONNELLE 1-N pour TACHE.

Association REALISATION-PROJET.

Description	Un projet se rattache à une ou plusieurs unités organisationnelles.
Liaison entre	UNITE ORGANISATIONNELLE et PROJET
Rôle	"contribue à" pour UNITE ORGANSATIONNELLE "se rattache à" pour PROJET
Connectivité	0-N pour UNITE ORGANISATIONNELLE 0-N pour PROJET.

Association RESPONSABLE-TACHE.

Description	Une ou plusieurs personnes sont responsables d'une tâche. En cas de problème, on fait appel à ces personnes.
Liaison entre	PERSONNE et TACHE.
Rôle	"est responsable de" pour PERSONNE "a pour responsable" pour TACHE.
Connectivité	0-N pour PERSONNE 1-N pour TACHE.

Association RESPONSABLE-PROJET.

Description	Une ou plusieurs personnes sont responsables d'un projet. En cas de problème, on fait appel à ces personnes.
Liaison entre	PERSONNE et PROJET.
Rôle	"est responsable de" pour PERSONNE "a pour responsable" pour PROJET.
Connectivité	0-N pour PERSONNE 1-N pour PROJET.

Association ACCOMPLISSEMENT.

Description	Une tâche est réalisée par une ou plusieurs personnes.
Liaison entre	TACHE et PERSONNE.
Rôle	"exécute" pour PERSONNE "est exécutée par" pour TACHE.
Connectivité	0-N pour PERSONNE 0-N pour TACHE.

Association COMPOSE-PROJET.

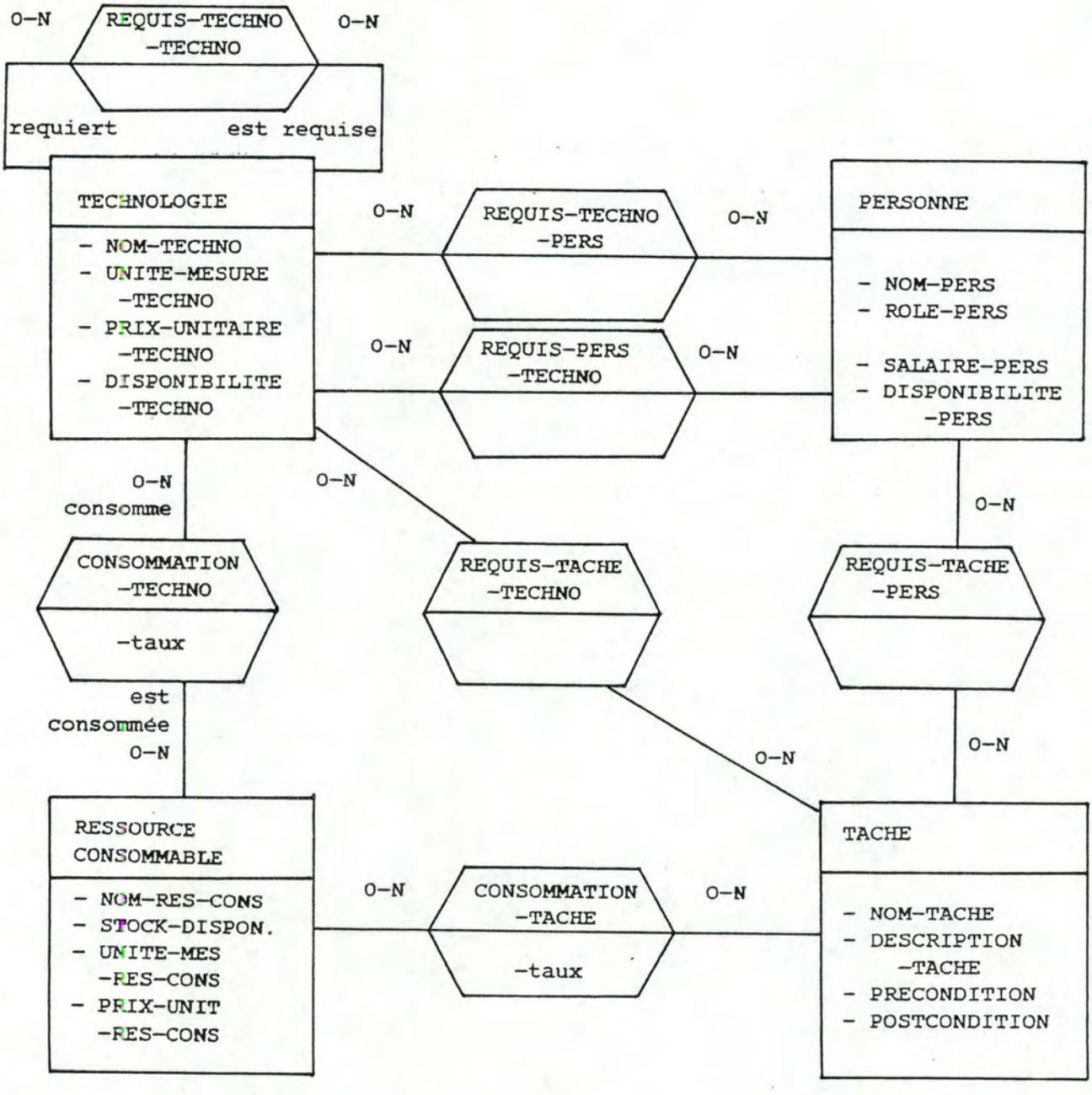
Description	Un projet se compose de tâches temporaires.
Liaison entre	PROJET et TACHE.
Rôle	"se compose de" pour PROJET "fait partie de" pour TACHE.
Connectivité	1-N pour PROJET 0-N pour TACHE.

3.2.4 LE MODELE DE STRUCTURATION DES RESSOURCES

Ce modèle permet de définir les ressources nécessaires pour exécuter les tâches de bureau décrites par le modèle organisationnel.

Parmi les ressources, on distingue:
- les ressources consommables;
- les ressources réutilisables:
  - la technologie;
  - les personnes.

3.2.4.1 REPRESENTATION GRAPHIQUE



3.2.4.2 DESCRIPTION DES ENTITES

A.LES RESSOURCES REUTILISABLES.

Les personnes et la technologie constituent les ressources réutilisables; c'est-à-dire, des ressources qui, dès qu'est terminée la tâche pour laquelle elles ont été requises, sont disponibles pour une autre tâche.

Entité TECHNOLOGIE.

a. Définition.

C'est l'ensemble du matériel (informatique ou non) et des logiciels qui exécutent totalement ou partiellement une tâche.

b. Attributs.

- NOM-TECHNOLOGIE.
  Définition:
  C'est le nom de la technologie.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- UNITE-MESURE-TECHNO.
  Définition:
  Unité de mesure de la technologie.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.
  Exemple: "kilo-octets" pour la mémoire.

- PRIX-UNITAIRE-TECHNO.
  Définition:
  Prix par unité de temps.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un entier.

- DISPONIBILITE-TECHNO.
  C'est un attribut décomposable selon les aspects:
  - la CAPACITE GLOBALE, exprimée en unités de mesure de la ressource réutilisable;
  - le CALENDRIER DE DISPONIBILITE, donnant les moments de disponibilité;
  - le NOMBRE DE POINTS D'ENTREE, donnant le nombre de tâches ou de ressources réutilisables qui peuvent se partager, dans les limites de la capacité, l'utilisation de la ressource réutilisable.

c. Identifiant.

Une technologie est identifiée par son nom NOM-TECHNOLOGIE.

Entité PERSONNE.

Ici, on considère les personnes en tant que facteurs de production et non plus comme membres de l'organisation. En plus des attributs déjà cités dans le modèle organisationnel, on ajoutera les attributs suivants afin de tenir compte du fait qu'une personne est une ressource de l'organisation:

- SALAIRE-PERS.
  Définition:
  C'est le prix de la ressource PERSONNE.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un entier.

- DISPONIBILITE-PERS.
  C'est un attribut décomposable selon les aspects:
  - la CAPACITE GLOBALE, exprimée en unités de mesure de la ressource réutilisable;
  - le CALENDRIER DE DISPONIBILITE, donnant les moments de disponibilité;
  - le NOMBRE DE POINTS D'ENTREE, donnant le nombre de tâches ou de ressources réutilisables qui peuvent se partager, dans les limites de la capacité, l'utilisation de la ressource réutilisable.

B. LES RESSOURCES CONSOMMABLES.

Entité RESSOURCE-CONSOMMABLE.

a. Définition.

C'est une ressource qui n'est pas réutilisable lorsqu'elle a été consommée lors de l'exécution d'une tâche. Exemples: papier, énergie, argent.

b. Attributs.

- NOM-RES-CONS.
  Définition:
  C'est le nom de la ressource consommable.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- UNITE-MESURE-RES-CONS.
  Définition:
  Unité de mesure de la ressource consommable.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- PRIX-UNITAIRE-RES-CONS.
  Définition:
  Prix par unité de temps.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un entier.

- STOCK-DISPONIBLE.
  Définition:
  Stock disponible de la ressource consommable au cours d'un laps de temps.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.

c. Identifiant.

Une ressource consommable est identifiée par son nom NOM-RES-CONS.

3.2.4.3 DESCRIPTION DES ASSOCIATIONS

Association REQUISITION-TACHE-TECHNO.

Description	Une tâche peut requérir de la technologie pour son exécution.
Liaison entre	TACHE et TECHNOLOGIE.
Rôle	"requiert" pour TACHE "est requise" pour TECHNOLOGIE.
Connectivité	0-N pour TACHE 0-N pour TECHNOLOGIE.
Attribut	QTE-REQUISE: quantité de technologie que requiert la tâche.

Association REQUISITION-TACHE-PERSONNE.

Description	Une tâche peut requérir des personnes pour son exécution.
Liaison entre	TACHE et PERSONNE.
Rôle	"requiert" pour TACHE "est requise" pour PERSONNE.
Connectivité	0-N pour TACHE 0-N pour TECHNOLOGIE.
Attribut	QTE-REQUISE: quantité de personnes que requiert la tâche.

Association REQUISITION-TECHNO-TECHNO.

Description	Une technologie peut requérir une autre technologie pour l'exécution d'une tâche.
Liaison entre	TECHNOLOGIE et TECHNOLOGIE.
Rôle	"requiert" pour TECHNOLOGIE "est requise" pour TECHNOLOGIE.
Connectivité	0-N pour TECHNOLOGIE 0-N pour TECHNOLOGIE.
Attribut	QTE-REQUISE: quantité de technologie que requiert la technologie.

Association REQUISITION-TECHNO-PERS.

Description	Une technologie peut nécessiter la réquisition d'une personne dans l'exécution d'une tâche.
Liaison entre	PERSONNE et TECHNOLOGIE.
Rôle	"requiert" pour TECHNOLOGIE "est requise" pour PERSONNE.
Connectivité	0-N pour PERSONNE 0-N pour TECHNOLOGIE.
Attribut	QTE-REQUISE: quantité de personnel que requiert la technologie.

Association REQUISITION-PERS-TECHNO.

Description	Une personne peut nécessiter la réquisition d'une technologie dans l'exécution d'une tâche.
Liaison entre	PERSONNE et TECHNOLOGIE.
Rôle	"requiert" pour PERSONNE "est requise" pour TECHNOLOGIE.
Connectivité	0-N pour PERSONNE 0-N pour TECHNOLOGIE.
Attribut	QTE-REQUISE: quantité de technologie que requiert la personne.

Association CONSOMMATION-TACHE.

Description	Une tâche peut consommer des ressources consommables pour son exécution.
Liaison entre	TACHE et RESSOURCE-CONSOMMABLE.
Rôle	"consomme" pour TACHE "est consommée" pour RESSOURCE-CONSOMMABLE.
Connectivité	0-N pour TACHE 0-N pour RESSOURCE-CONSOMMABLE.
Attribut	TAUX: taux de consommation.

Association CONSOMMATION-TECHNO.

Description	Une technologie peut consommer des ressources consommables dans l'exécution d'une tâche.
Liaison entre	TECHNOLOGIE et RESSOURCE-CONSOMMABLE.
Rôle	"consomme" pour TECHNOLOGIE "est consommée" pour RESSOURCE-CONSOMMABLE.
Connectivité	0-N pour TECHNOLOGIE 0-N pour RESSOURCE-CONSOMMABLE.
Attribut	TAUX: taux de consommation.

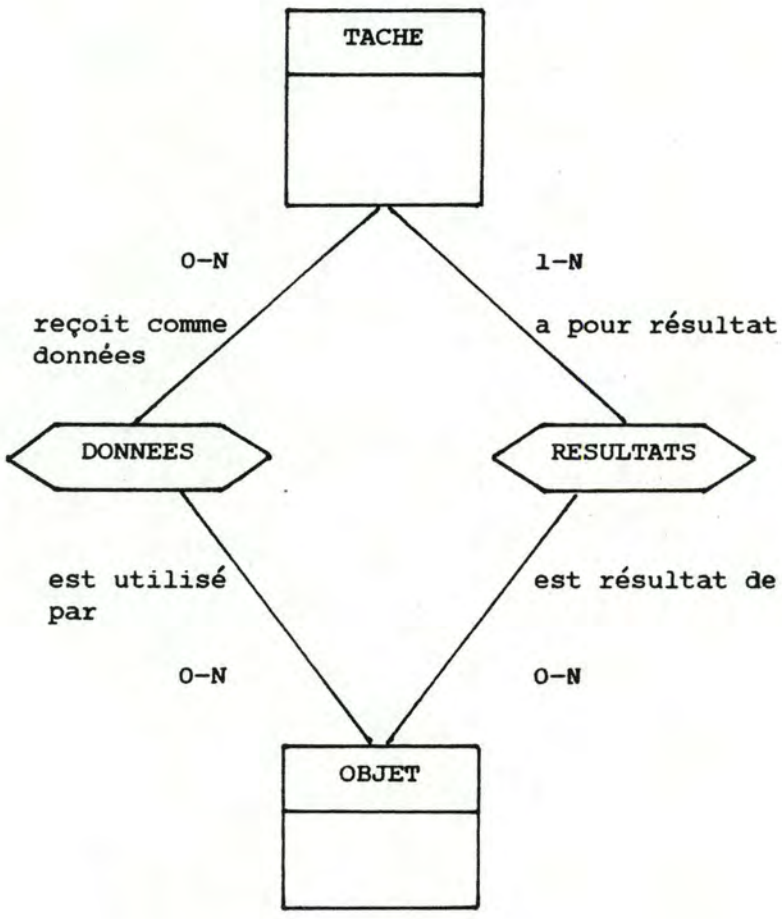
3.2.5 LE MODELE DE STRUCTURATION DE L'INFORMATION

Comme nous l'avons déjà signalé, dans le cadre du bureau, nous envisageons des tâches qui traitent de l'information. En bureautique, six objets suffisent à décrire les données manipulées. Ce sont le message, le document, le formulaire, le dossier, le fichier et la pile.

Le message, le document et le formulaire sont des objets élémentaires qui permettent de représenter l'information manipulée dans les bureaux.

Le dossier, le fichier et la pile sont des objets structurants. Ils permettent d'organiser les informations, de les grouper en vue de leur utilisation ultérieure.

Les tâches reçoivent des objets-données. Elles les modifient ou se servent de l'information qu'ils véhiculent pour en générer d'autres. On peut représenter cela graphiquement comme suit:



- L'entité TACHE représente une des tâches décrites dans le modèle organisationnel.
- L'entité OBJET représente un message, un document, un formulaire, un dossier, un fichier ou une pile.
- Les associations DONNEES et RESULTATS décrivent les manipulations de l'information dans les bureaux.

Le modèle de structuration de l'information va nous permettre de formaliser les différents objets.

3.2.5.1 DESCRIPTION GENERALE DES OBJETS

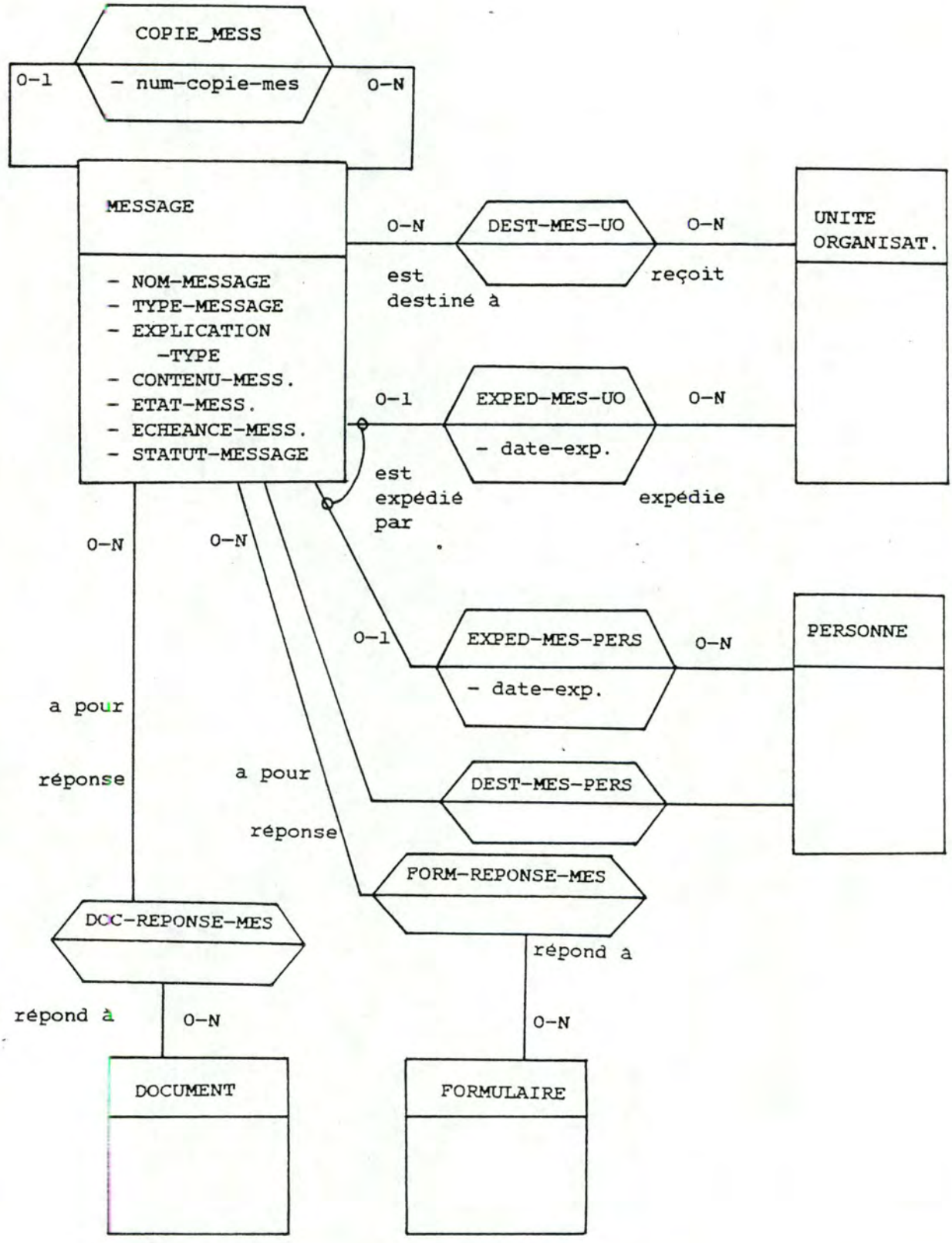
Tout objet informationnel sera décrit par une entité. On peut grouper les attributs de cette entité selon deux types:

- des attributs "statiques" :
  Ils décrivent l'objet, son contenu, ... Ces attributs ne changent pas au cours de la vie de l'objet.

- des attributs "dynamiques" :
  Ils peuvent varier au cours de la vie de l'objet. Ces attributs sont régis par un attribut particulier: l'ETAT de l'objet. L'état permet de décrire l'évolution dynamique, au cours du temps, de l'objet. L'état est modifié lorsque l'on manipule l'objet et un changement d'état de l'objet peut déclencher un nouveau traitement de celui-ci ou le traitement d'un autre objet.

L'entité représentant un objet informationnel sera mise en relation avec les objets du modèle organisationnel afin de décrire l'origine et la destination de l'information dans l'organisation.

3.2.5.2 DESCRIPTION DU MESSAGE



Entité MESSAGE.

a. Définition.

Un message est une information sous forme écrite / orale , qui a les caractéristiques suivantes:
- elle est en général brève;
- le corps n'obéit à aucune structure particulière;
- le moyen de communication utilisé est généralement synchrone (téléphone, réunion).

b. Attributs.

- NOM-MESSAGE.
  Définition:
  C'est le nom du message.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- TYPE-MESSAGE.
  Définition:
  C'est le type du message.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères qui peut prendre les valeurs: coup de téléphone, mémo, réunion, telex, télégramme, avis, notice administrative, calendrier ou autre.

- EXPLICATION-TYPE.
  Définition:
  Si le type du message est "autre", l'attribut EXPLICATION-TYPE permet à l'utilisateur d'en donner une brève description.
  Propriétés:
  C'est un attribut simple, élémentaire et facultatif.
  Format:
  C'est un texte.

- CONTENU-MESSAGE.
  Définition:
  C'est le corps du message.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est un texte.

- ETAT-MESSAGE.
  Définition:
    Concept relatif au cycle de vie du message.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite finie de caractères.

- ECHEANCE-MESSAGE.
  Définition:
    C'est la date à laquelle le message sera traité au plus tard.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une date.

- STATUT-MESSAGE.
  Définition:
    Cet attribut permet de savoir si le message est un original ou une copie.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères qui peut prendre les valeurs: ORIGINAL ou COPIE.

c. Identifiant.

Un message est identifié par son nom NOM-MESSAGE et par son numéro de copie NUM-COPIE-MES.

Autres entités.

Les entités PERSONNE et UNITE-ORGANISATIONNELLE sont décrites dans le modèle organisationnel.
Les entités DOCUMENT et FORMULAIRE sont définies ci-après.

Association EXPED-MES-PERS.

Description	L'expéditeur d'un message peut être une personne.
Liaison entre	MESSAGE et PERSONNE.
Rôle	"est expédié par" pour MESSAGE "expédie" pour PERSONNE.
Connectivité	0-1 pour MESSAGE 0-N pour PERSONNE.
Attribut	DATE-EXPEDITION: date d'expédition du message.

Association EXPED-MES-UO.

Description	L'expéditeur d'un message peut être une unité organisationnelle.
Liaison entre	MESSAGE et UNITE-ORGANISATIONNELLE.
Rôle	"est expédié par" pour MESSAGE "expédie" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-1 pour MESSAGE 0-N pour UNITE-ORGANISATIONNELLE.
Attribut	DATE-EXPEDITION: date d'expédition du message.

Association DEST-MES-PERS.

Description	Le destinataire d'un message peut être une personne.
Liaison entre	MESSAGE et PERSONNE.
Rôle	"est destiné à" pour MESSAGE "reçoit" pour PERSONNE.
Connectivité	0-N pour MESSAGE 0-N pour PERSONNE.

Association DEST-MES-UO.

Description	Le destinataire d'un message peut être une unité organisationnelle.
Liaison entre	MESSAGE et UNITE-ORGANISATIONNELLE.
Rôle	"est destiné à" pour MESSAGE "reçoit" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-N pour MESSAGE 0-N pour UNITE-ORGANISATIONNELLE.

Contraintes d'intégrité.

Un message peut être envoyé simultanément à une ou plusieurs unités organisationnelles et/ou une ou plusieurs personnes.

(exclusion) L'expéditeur du message est soit une personne, soit une unité organisationnelle.

Association DOC-REPONSE-MES.

Description	Un message peut requérir un ou plusieurs documents en réponse.
Liaison entre	MESSAGE et DOCUMENT.
Rôle	"a pour réponse" pour MESSAGE "répond à" pour DOCUMENT.
Connectivité	0-N pour MESSAGE 0-N pour DOCUMENT.

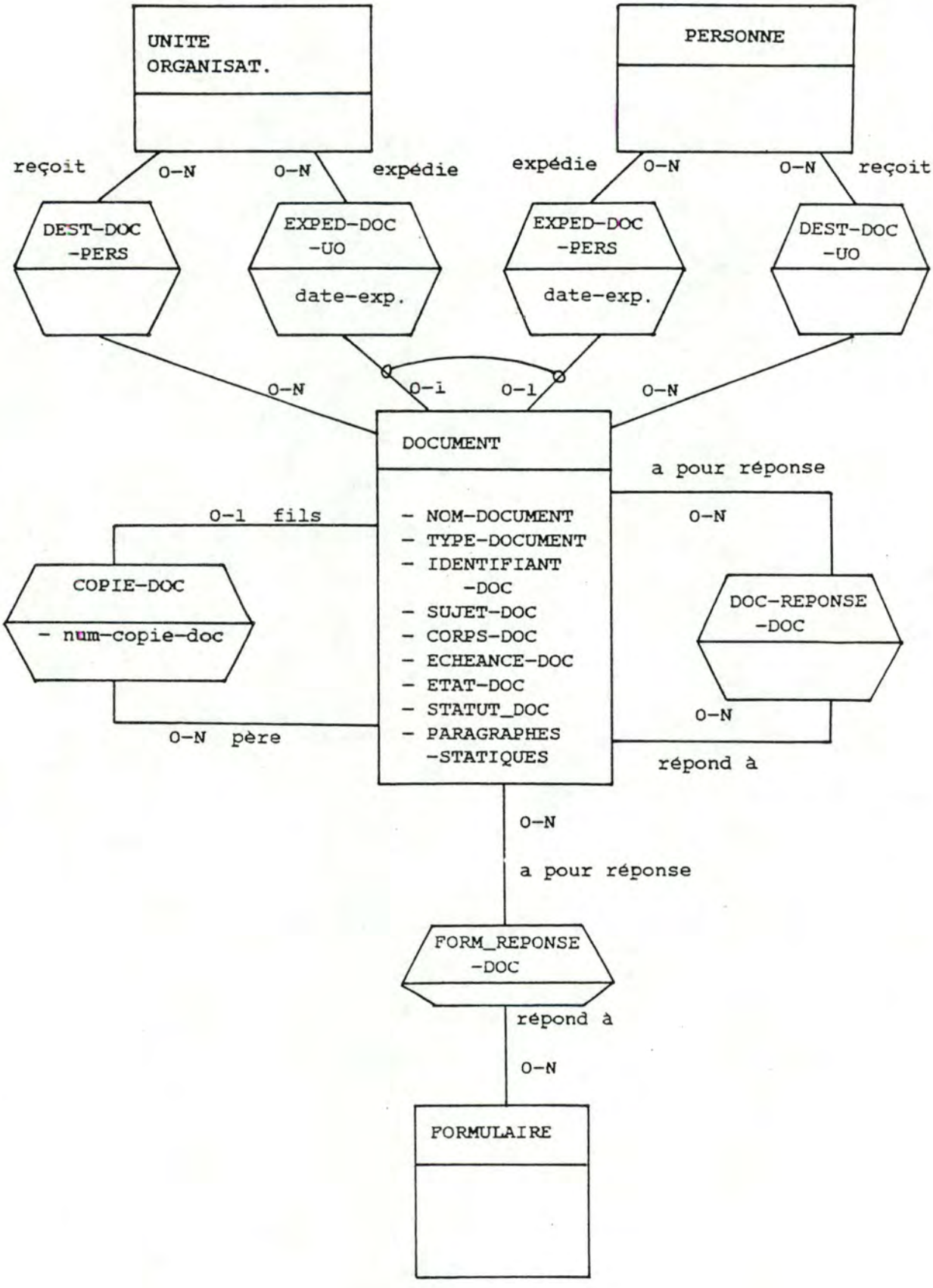
Association FORM-REPONSE-MES.

Description	Un message peut requérir un ou plusieurs formulaires en réponse.
Liaison entre	MESSAGE et FORMULAIRE
Rôle	"a pour réponse" pour MESSAGE "répond à" pour FORMULAIRE.
Connectivité	0-N pour MESSAGE 0-N pour FORMULAIRE.

Association COPIE-MES.

Description	Un message peut être une copie d'un autre.
Liaison entre	MESSAGE (1) et MESSAGE (2).
Rôle	"est copie de" pour MESSAGE (1) "est original de" pour MESSAGE (2).
Connectivité	0-1 pour MESSAGE (1) 0-N pour MESSAGE (2).
Attribut	NUM-COPIE-MES: numéro de copie du message

3.2.5.3 DESCRIPTION DU DOCUMENT



Entité DOCUMENT.

1. Définition.

Un document est une information spécifique, généralement représentée sous forme écrite, graphique ou mixte. Les documents demandent à être produits, sont généralement classés / archivés et leur distribution se fait selon des moyens de communication asynchrones (poste interne ou externe).

2. Attributs.

- NOM-DOCUMENT.
  Définition:
  C'est le nom du document.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une chaine de caractères.

- TYPE-DOCUMENT.
  Définition:
  C'est le type du document.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite de caractères qui peut prendre les valeurs: lettre, brochure, journal, article, rapport de réunion, livre, projet (d'un contrat), autre.

- EXPLICATION-TYPE.
  Définition:
  Si le type du document est "autre", l'attribut EXPLICATION-TYPE permet à l'utilisateur d'en donner une brève description.
  Propriétés:
  C'est un attribut simple, élémentaire et facultatif.
  Format:
  C'est un texte.

- IDENTIFIANT-DOC.
  Définition:
  C'est l'identifiant du document.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- SUJET-DOC.
  Définition:
    C'est une description concise, sous forme de mots-clé, de l'objet du document.
  Propriétés:
    C'est un attribut répétitif (un ou plusieurs mots-clé), élémentaire et obligatoire.
  Format:
    C'est une suite finie de caractères.

- CORPS-DOC.
  Définition:
    C'est le texte constituant le document.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est un texte.

- STATUT-DOC.
  Définition:
    Cet attribut permet de savoir si le document est un original ou une copie.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères qui peut prendre les valeurs: ORIGINAL ou COPIE.

- ECHEANCE-DOC.
  Définition:
    C'est la date à laquelle le document sera traité au plus tard.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une date.

- ETAT-DOC.
  Définition:
    Concept lié au cycle de vie du document.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite finie de caractères.

- PAR-STATIQUES.
  Définition:
    Cet attribut dit si le document contient des paragraphes statiques.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est un booléen.

3. Identifiant.

Un document est identifié par son identifiant IDENTIFIANT-DOC et par son numéro de copie NUM-COPIE-DOC.

Association EXPED-DOC-PERS.

Description	L'expéditeur d'un document peut être une personne.
Liaison entre	DOCUMENT et PERSONNE.
Rôle	"est expédié par" pour DOCUMENT "expédie" pour PERSONNE.
Connectivité	0-1 pour DOCUMENT 0-N pour PERSONNE.
Attribut	DATE-EXPEDITION: date d'expédition du document.

Association EXPED-DOC-UO.

Description	L'expéditeur d'un document peut être une unité organisationnelle.
Liaison entre	DOCUMENT et UNITE-ORGANISATIONNELLE.
Rôle	"est expédié par" pour DOCUMENT "expédie" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-1 pour DOCUMENT 0-N pour UNITE-ORGANISATIONNELLE.
Attribut	DATE-EXPEDITION: date d'expédition du document.

Association DEST-DOC-PERS.

Description	Le destinataire d'un document peut être une personne.
Liaison entre	DOCUMENT et PERSONNE.
Rôle	"est destiné à" pour DOCUMENT "reçoit" pour PERSONNE.
Connectivité	0-N pour DOCUMENT 0-N pour PERSONNE.

Association DEST-DOC-UO.

Description	Le destinataire d'un document peut être une unité organisationnelle.
Liaison entre	DOCUMENT et UNITE-ORGANISATIONNELLE.
Rôle	"est destiné à" pour DOCUMENT "reçoit" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-N pour DOCUMENT 0-N pour UNITE-ORGANISATIONNELLE.

Contraintes d'intégrité.

Un document peut être envoyé simultanément à une ou plusieurs unités organisationnelles et/ou une ou plusieurs personnes.

(exclusion) L'expéditeur du document est soit une personne, soit une unité organisationnelle.

Association DOC-REPONSE-DOC.

Description	Un document peut requérir un ou plusieurs documents en réponse.
Liaison entre	DOCUMENT et DOCUMENT.
Rôle	"a pour réponse" pour DOCUMENT "répond à" pour DOCUMENT.
Connectivité	0-N pour DOCUMENT 0-N pour DOCUMENT.

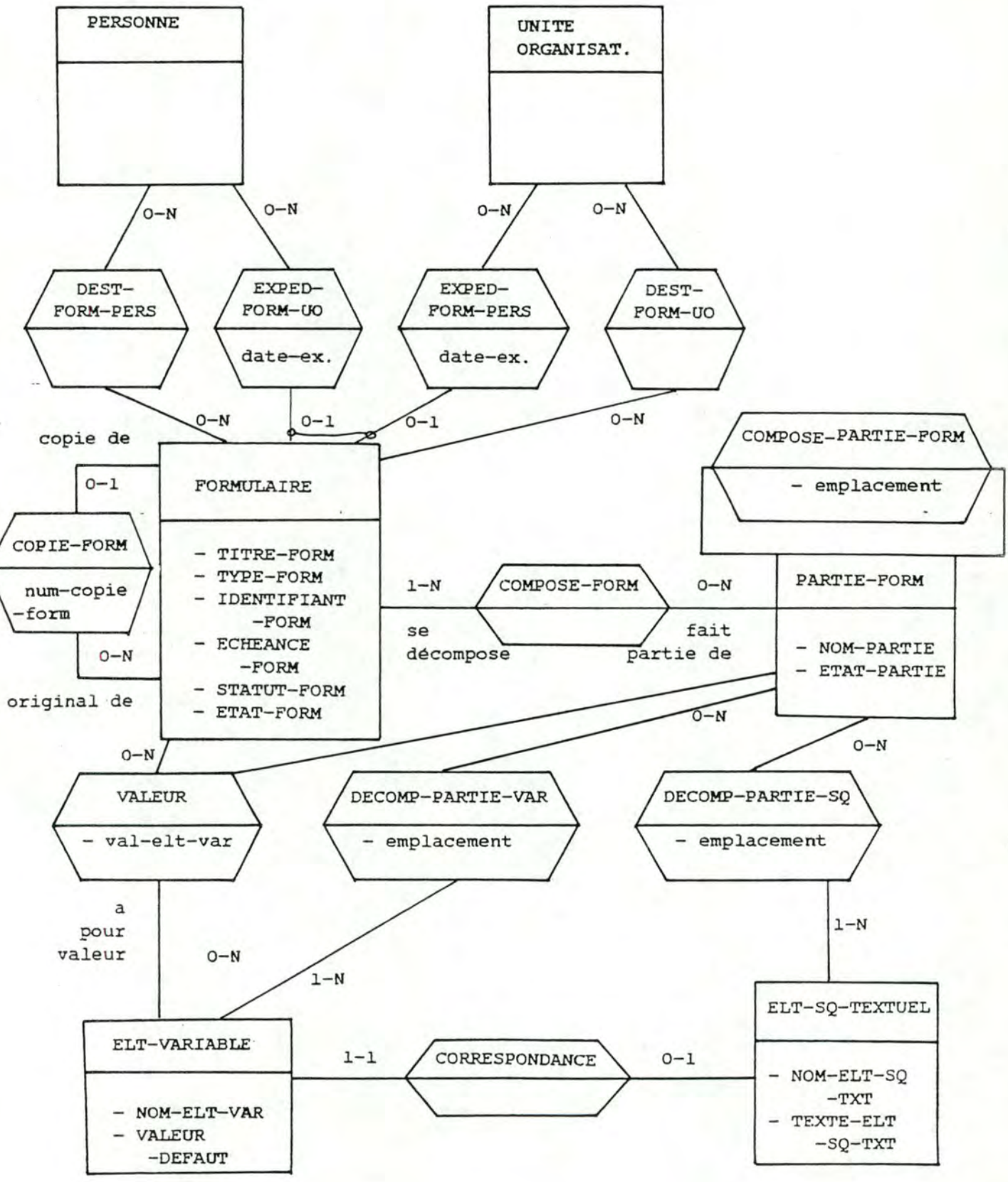
Association FORM-REPONSE-DOC.

Description	Un document peut requérir un ou plusieurs formulaires en réponse.
Liaison entre	DOCUMENT et FORMULAIRE
Rôle	"a pour réponse" pour DOCUMENT "répond à" pour FORMULAIRE.
Connectivité	0-N pour DOCUMENT 0-N pour FORMULAIRE.

Association COPIE-DOC.

Description	Un document peut être une copie d'un autre.
Liaison entre	DOCUMENT (1) et DOCUMENT (2).
Rôle	"est copie de" pour DOCUMENT (1) "est original de" pour DOCUMENT (2).
Connectivité	0-1 pour DOCUMENT (1) 0-N pour DOCUMENT (2).
Attribut	NUM-COPIE-DOC: numéro de copie du document.

3.2.5.4 DESCRIPTION DU FORMULAIRE



Entité FORMULAIRE.

a. Définition.

Un formulaire est une information écrite standardisée. Il est composé d'un squelette textuel et d'éléments auxquels peuvent être associés des ensembles de valeurs.

b. Attributs.

- TITRE-FORM.
  Définition:
  C'est le titre que porte le formulaire.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- TYPE-FORM.
  Définition:
  Cet attribut dit s'il s'agit d'un véritable formulaire ou d'une liste (qui est un formulaire dont le squelette textuel est réduit).
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite de caractères qui peut prendre les valeurs: FORMULAIRE ou LISTE.

- IDENTIFIANT-FORM.
  Définition:
  Cet attribut permet d'identifier le formulaire.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ECHEANCE-FORM.
  Définition:
  C'est la date à laquelle le formulaire sera traité au plus tard.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une date.

- STATUT-FORM.
  Définition:
    Cet attribut dit si le formulaire est un original ou une copie.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères qui peut prendre les valeurs: ORIGINAL ou COPIE.

- ETAT-FORM.
  Définition:
    Concept lié au cycle de vie du formulaire.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères.

c. Identifiant.

Un formulaire est identifié par son nom TITRE-FORM et par son identifiant IDENTIFIANT-FORM et son numéro de copie NUM-COPIE-FORM.

Entité PARTIE-FORM.

a. Définition.

C'est une partie du formulaire. Chaque partie du formulaire possède une caractéristique propre, par exemple: telle partie est remplie par telle personne; telle partie traite de tel sujet; ... .

b. Attributs.

- NOM-PARTIE.
  Définition:
    C'est le nom de la partie du formulaire. Si possible, on choisira le nom de façon à refléter la caractéristique de la partie.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères.

- ETAT-PARTIE.
  Définition:
    Concept lié au cycle de vie de la partie du formulaire.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères.

c. Identifiant.

Une partie de formulaire est identifiée par son nom NOM-PARTIE et le nom TITRE-FORM du formulaire dont elle fait partie.

Entité ELT-SQ-TEXTUEL.

a. Définition.

C'est un élément du squelette textuel du formulaire.

b. Attributs.

- NOM-ELT-SQ-TEXTUEL.
  Définition:
    C'est le nom de l'élément du squelette textuel du formulaire.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères.

- TEXTE-ELT-SQ-TEXTUEL.
  Définition:
    C'est la valeur de l'élément du squelette textuel du formulaire.
  Propriétés:
    C'est un attribut simple, élémentaire et obligatoire.
  Format:
    C'est une suite de caractères.

c. Identifiant.

Un élément du squelette textuel du formulaire est identifié par son nom NOM-ELT-SQ-TEXTUEL ainsi que par les noms TITRE-FORM et NOM-PARTIE du formulaire et de la partie de formulaire auxquels il appartient.

Entité ELT-VARIABLE.

a. Définition.

C'est un élément à remplir du formulaire.

b. Attributs.

- NOM-ELT-VARIABLE.
  Définition:
  C'est le nom de l'élément variable.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite de caractères.

- VALEUR-DEFAUT.
  Définition:
  C'est la valeur par défaut de l'élément variable.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite de caractères.

c. Identifiant.

Un élément variable est identifié par son nom NOM-ELT-VARIABLE ainsi que par les noms TITRE-FORM et NOM-PARTIE du formulaire et de la partie de formulaire auxquels il appartient.

Association EXPED-FORM-PERS.

Description	L'expéditeur d'un formulaire peut être une personne.
Liaison entre	FORMULAIRE et PERSONNE.
Rôle	"est expédié par" pour FORMULAIRE "expédie" pour PERSONNE.
Connectivité	0-1 pour FORMULAIRE 0-N pour PERSONNE.
Attribut	DATE-EXPEDITION: date d'expédition du formulaire.

Association EXPED-FORM-UO.

Description	L'expéditeur d'un formulaire peut être une unité organisationnelle.
Liaison entre	FORMULAIRE et UNITE-ORGANISATIONNELLE.
Rôle	"est expédié par" pour FORMULAIRE "expédie" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-1 pour FORMULAIRE 0-N pour UNITE-ORGANISATIONNELLE.
Attribut	DATE-EXPEDITION: date d'expédition du formulaire.

Association DEST-FORM-PERS.

Description	Le destinataire d'un formulaire peut être une personne.
Liaison entre	FORMULAIRE et PERSONNE.
Rôle	"est destiné à" pour FORMULAIRE "reçoit" pour PERSONNE.
Connectivité	0-N pour FORMULAIRE 0-N pour PERSONNE.

Association DEST-FORM-UO.

Description	Le destinataire d'un formulaire peut être une unité organisationnelle.
Liaison entre	FORMULAIRE et UNITE-ORGANISATIONNELLE.
Rôle	"est destiné à" pour FORMULAIRE "reçoit" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-N pour FORMULAIRE 0-N pour UNITE-ORGANISATIONNELLE.

Contraintes d'intégrité.

Un formulaire peut être envoyé simultanément à une ou plusieurs unités organisationnelles et/ou une ou plusieurs personnes.

(exclusion) L'expéditeur du formulaire est soit une personne, soit une unité organisationnelle.

Association DOC-REPONSE-FORM.

Description	Un formulaire peut requérir un ou plusieurs documents en réponse.
Liaison entre	FORMULAIRE et DOCUMENT.
Rôle	"a pour réponse" pour FORMULAIRE "répond à" pour DOCUMENT.
Connectivité	0-N pour FORMULAIRE 0-N pour DOCUMENT.

Association FORM-REPONSE-FORM.

Description	Un formulaire peut requérir un ou plusieurs formulaires en réponse.
Liaison entre	FORMULAIRE et FORMULAIRE
Rôle	"a pour réponse" pour FORMULAIRE "répond à" pour FORMULAIRE.
Connectivité	0-N pour FORMULAIRE 0-N pour FORMULAIRE.

Association COPIE-FORM.

Description	Un formulaire peut être une copie d'un autre.
Liaison entre	FORMULAIRE(1) et FORMULAIRE(2).
Rôle	"est copie de" pour FORMULAIRE (1) "est original de" pour FORMULAIRE (2).
Connectivité	0-1 pour FORMULAIRE (1) 0-N pour FORMULAIRE (2).
Attribut	NUM-COPIE-FORM: numéro de copie du formulaire.

Association COMPOSE-FORM.

Description	Un formulaire se compose d'une ou plusieurs parties.
Liaison entre	FORMULAIRE et PARTIE-FORM.
Rôle	"se décompose en" pour FORMULAIRE "fait partie de pour PARTIE-FORM.
Connectivité	1-N pour FORMULAIRE 0-N pour PARTIE-FORM.

Association COMPOSE-PARTIE-FORM.

Description	Une partie de formulaire peut se décomposer plus finement en plusieurs autres parties.
Liaison entre	PARTIE-FORM et PARTIE-FORM.
Rôle	"se décompose en" pour PARTIE-FORM "fait partie de" pour PARTIE-FORM.
Connectivité	0-N pour PARTIE-FORM 0-N pour PARTIE-FORM.

Association DECOMP-PARTIE-SQ.

Description	La décomposition la plus fine est celle en éléments du squelette textuel et éléments variables.
Liaison entre	PARTIE-FORM et ELT-SQ-TEXTUEL.
Rôle	"se décompose en" pour PARTIE-FORM "fait partie de" pour ELT-SQ-TEXTUEL.
Connectivité	0-N pour PARTIE-FORM 1-N pour ELT-SQ-TEXTUEL.

Association DECOMP-PARTIE-VAR.

Description	La décomposition la plus fine est celle en éléments du squelette textuel et éléments variables.
Liaison entre	PARTIE-FORM et ELT-VARIABLE.
Rôle	"se décompose en" pour PARTIE-FORM "fait partie de" pour ELT-VARIABLE.
Connectivité	0-N pour PARTIE-FORM 1-N pour ELT-VARIABLE.

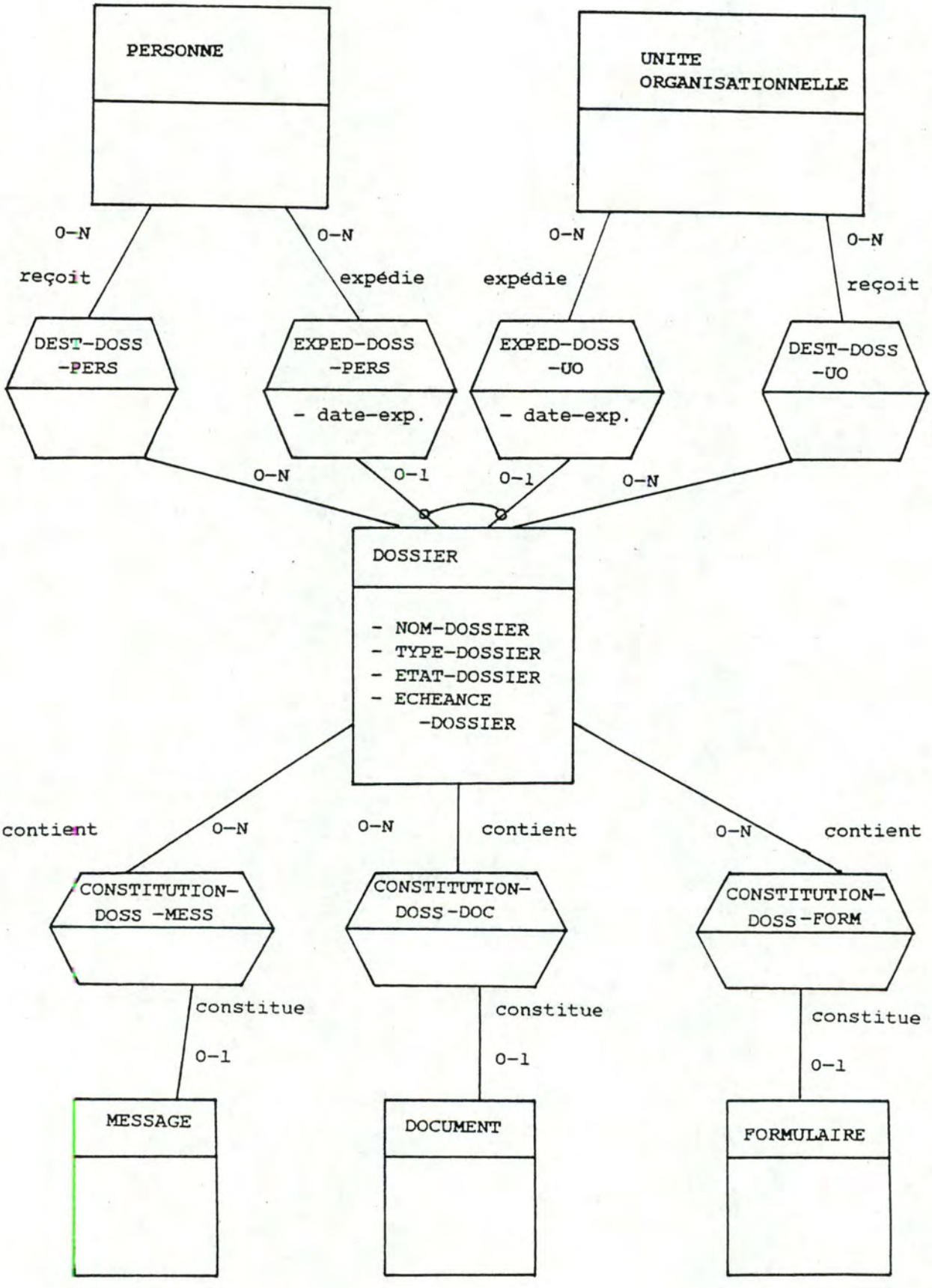
Association CORRESPONDANCE.

Description	Un élément variable correspond à un élément du squelette textuel.
Liaison entre	ELT-VARIABLE et ELT-SQ-TEXTUEL.
Rôle	"correspond à" pour ELT-VARIABLE "a pour valeur" pour ELT-SQ-TEXTUEL.
Connectivité	1-1 pour ELT-VARIABLE 0-1 pour ELT-SQ-TEXTUEL.

Association VALEUR.

Description	A chaque élément variable, on peut associer un ensemble de valeurs.
Liaison entre	FORMULAIRE, PARTIE-FORM et ELT-VARIABLE.
Rôle	"a pour valeur" pour ELT-VARIABLE.
Connectivité	0-N pour FORMULAIRE 0-N pour PARTIE-FORM 0-N pour ELT-VARIABLE.
Attribut	VAL-ELT-VAR: valeur de l'élément variable.

3.2.5.5 DESCRIPTION DU DOSSIER



Entité DOSSIER.

a. Définition.

Les dossiers sont des groupes de plusieurs types différents de formulaires / messages / documents reliés entre eux par une relation logique.

b. Attributs.

- NOM-DOSSIER.
  Définition:
  C'est le nom du dossier. Celui-ci reflètera la relation logique existant entre les constituants du dossier, par exemple: dossier Etudiant, dossier Maladie.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite de caractères.

- IDENTIFIANT-DOSSIER.
  Définition:
  C'est l'identifiant du dossier.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ETAT-DOSSIER.
  Définition:
  Concept lié au cycle de vie du dossier.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ECHEANCE-DOSSIER.
  Définition:
  C'est la date à laquelle le dossier sera traité au plus tard.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une date.

c. Identifiant.

Un dossier est identifié par son nom NOM-DOSSIER et par son identifiant IDENTIFIANT-DOSSIER.

Association EXPED-DOS-PERS.

Description	L'expéditeur d'un dossier peut être une personne.
Liaison entre	DOSSIER et PERSONNE.
Rôle	"est expédié par" pour DOSSIER "expédie" pour PERSONNE.
Connectivité	0-1 pour DOSSIER 0-N pour PERSONNE.
Attribut	DATE-EXPEDITION: date d'expédition du dossier.

Association EXPED-DOS-UO.

Description	L'expéditeur d'un dossier peut être une unité organisationnelle.
Liaison entre	DOSSIER et UNITE-ORGANISATIONNELLE.
Rôle	"est expédié par" pour DOSSIER "expédie" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-1 pour DOSSIER 0-N pour UNITE-ORGANISATIONNELLE.
Attribut	DATE-EXPEDITION: date d'expédition du dossier.

Association DEST-DOS-PERS.

Description	Le destinataire d'un dossier peut être une personne.
Liaison entre	DOSSIER et PERSONNE.
Rôle	"est destiné à" pour DOSSIER "reçoit" pour PERSONNE.
Connectivité	0-N pour DOSSIER 0-N pour PERSONNE.

Association DEST-DOS-UO.

Description	Le destinataire d'un dossier peut être une unité organisationnelle.
Liaison entre	DOSSIER et UNITE-ORGANISATIONNELLE.
Rôle	"est destiné à" pour DOSSIER "reçoit" pour UNITE-ORGANISATIONNELLE.
Connectivité	0-N pour DOSSIER 0-N pour UNITE-ORGANISATIONNELLE.

Contraintes d'intégrité

Un dossier peut être envoyé simultanément à une ou plusieurs personnes et/ou une ou plusieurs unités organisationnelles.

(exclusion) L'expéditeur du dossier est soit une personne, soit une unité organisationnelle.

Association CONSTITUTION-DOS-MES.

Description	Un dossier peut comprendre un ou plusieurs types de messages.
Liaison entre	DOSSIER et MESSAGE.
Rôle	"contient" pour DOSSIER "contenu dans" pour MESSAGE.
Connectivité	0-N pour DOSSIER 0-1 pour MESSAGE.

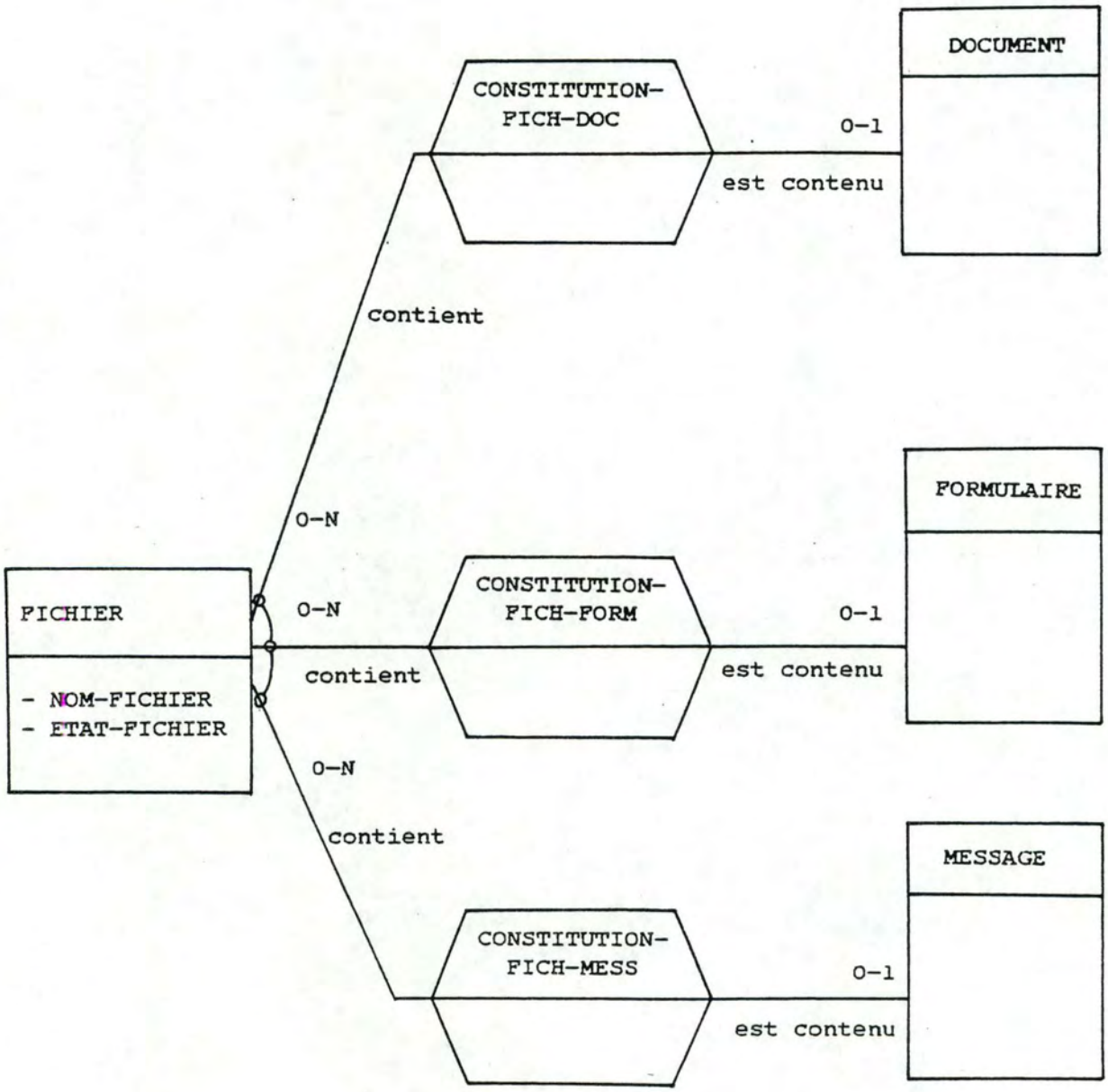
Association CONSTITUTION-DOS-DOC.

Description	Un dossier peut comprendre un ou plusieurs types de documents.
Liaison entre	DOSSIER et DOCUMENT.
Rôle	"contient" pour DOSSIER "contenu dans" pour DOCUMENT.
Connectivité	0-N pour DOSSIER 0-1 pour DOCUMENT.

Association CONSTITUTION-DOS-FORM.

Description	Un dossier peut comprendre un ou plusieurs types de formulaires.
Liaison entre	DOSSIER et FORMULAIRE.
Rôle	"contient" pour DOSSIER "contenu dans" pour FORMULAIRE.
Connectivité	0-N pour DOSSIER 0-1 pour FORMULAIRE.

3.2.5.6 DESCRIPTION DU FICHIER



Entité FICHIER.

1. Définition.

Les fichiers sont des groupes d'occurrences d'un même type de formulaires (quelquefois de messages ou de documents), univoquement identifiés et arrangés selon un certain ordre (alphabétique, chronologique). Tout fichier porte un identifiant.

2. Attributs.

- NOM-FICHIER.
  Définition:
  C'est le nom du fichier.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ETAT-FICHIER.
  Définition:
  Concept lié au cycle de vie du fichier.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

3. Identifiant.

Un fichier est identifié par son nom NOM-FICHIER.

Contrainte d'intégrité.

(exclusion) Un fichier est constitué d'un même type de messages ou d'un même type de documents ou d'un même type de formulaires.

Association CONSTITUTION-FICH-MES.

Description	Un fichier peut comprendre un seul type de messages.
Liaison entre	FICHIER et MESSAGE.
Rôle	"contient" pour FICHIER "contenu dans" pour MESSAGE.
Connectivité	0-N pour FICHIER 0-1 pour MESSAGE.

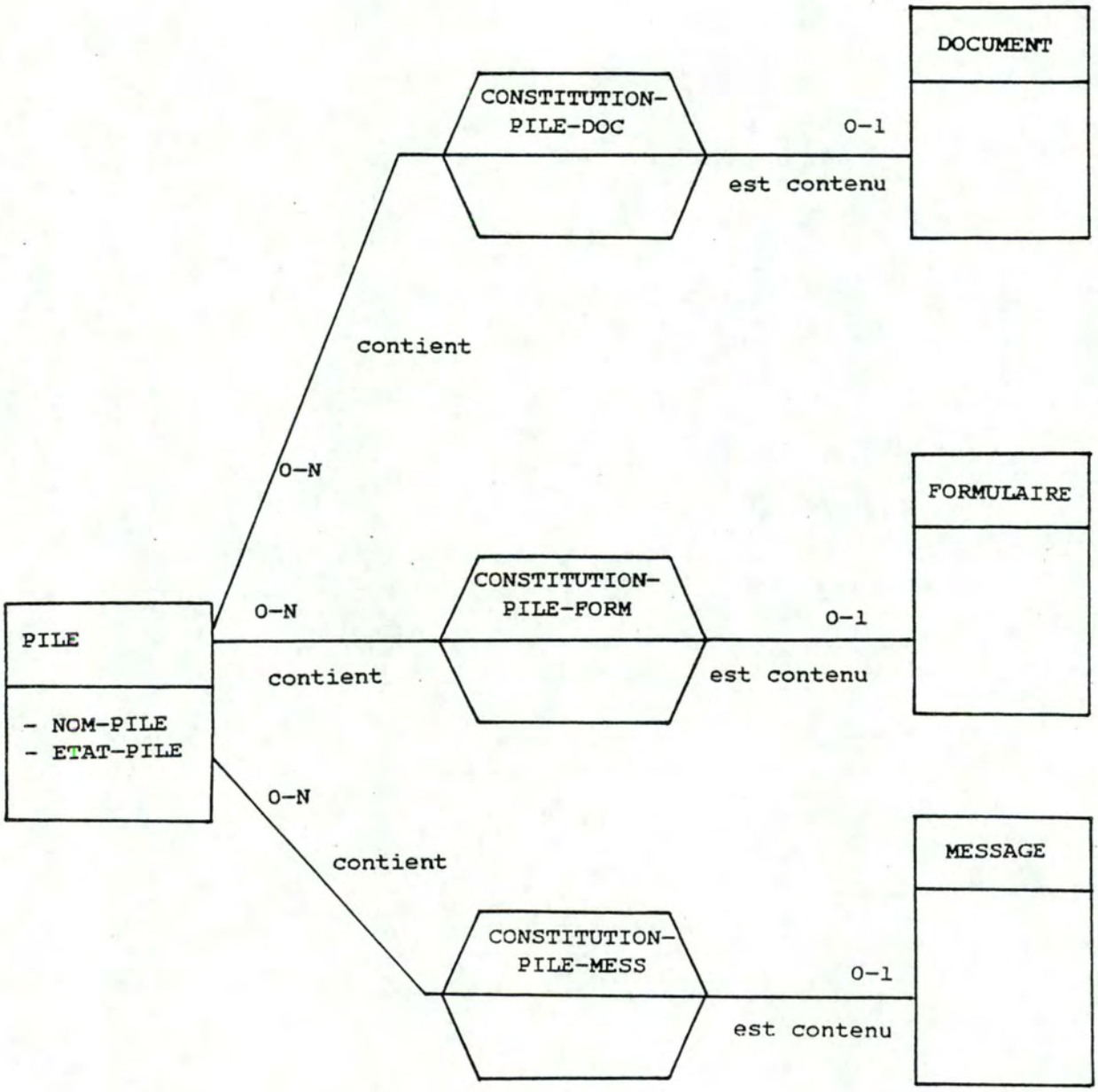
Association CONSTITUTION-FICH-DOC.

Description	Un fichier peut comprendre un seul type de documents.
Liaison entre	FICHIER et DOCUMENT.
Rôle	"contient" pour FICHIER "contenu dans" pour DOCUMENT.
Connectivité	0-N pour FICHIER 0-1 pour DOCUMENT.

Association CONSTITUTION-FICH-FORM.

Description	Un fichier peut comprendre un seul type de formulaires.
Liaison entre	FICHIER et FORMULAIRE.
Rôle	"contient" pour FICHIER "contenu dans" pour FORMULAIRE.
Connectivité	0-N pour FICHIER 0-1 pour FORMULAIRE.

3.2.5.7 DESCRIPTION DE LA PILE



Entité PILE.

a. Définition.

Les piles sont des groupes d'occurrences de divers types de documents / messages / formulaires, sans lien logique entre eux, sauf généralement l'ordre chronologique. Mais cet ordre n'est pas intentionnel. Ce sont des informations en attente de classement, de traitement.

b. Attributs.

- NOM-PILE.
  Définition:
  C'est le nom de la pile.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

- ETAT-PILE.
  Définition:
  Concept lié au cycle de vie de la pile.
  Propriétés:
  C'est un attribut simple, élémentaire et obligatoire.
  Format:
  C'est une suite finie de caractères.

c. Identifiant.

Une pile est identifiée par son nom NOM-PILE.

Association CONSTITUTION-PILE-MES.

Description	Une pile peut comprendre un ou plusieurs types de messages.
Liaison entre	PILE et MESSAGE.
Rôle	"contient" pour PILE "contenu dans" pour MESSAGE.
Connectivité	0-N pour PILE 0-1 pour MESSAGE.

Association CONSTITUTION-PILE-DOC.

Description	Une pile peut comprendre un ou plusieurs types de documents.
Liaison entre	PILE et DOCUMENT.
Rôle	"contient" pour PILE "contenu dans" pour DOCUMENT.
Connectivité	0-N pour PILE 0-1 pour DOCUMENT.

Association CONSTITUTION-PILE-FORM.

Description	Une pile peut comprendre un ou plusieurs types de formulaires.
Liaison entre	PILE et FORMULAIRE.
Rôle	"contient" pour PILE "contenu dans" pour FORMULAIRE.
Connectivité	0-N pour PILE 0-1 pour FORMULAIRE.

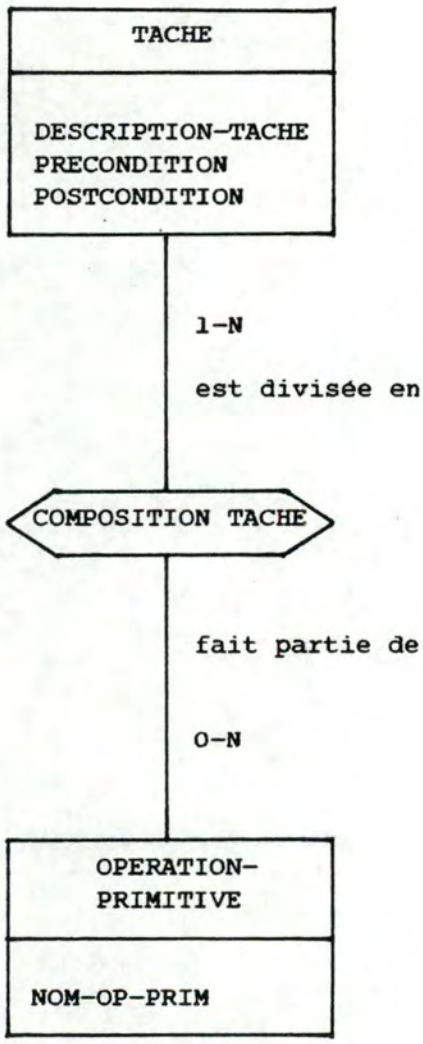
3.2.6 LE MODELE DE STRUCTURATION DES TRAITEMENTS

3.2.6.1 LA DECOMPOSITION DES TRAITEMENTS

Le travail de bureau peut être décomposé en différentes tâches qui remplissent chacune un sous-objectif pour atteindre le but du bureau.

Une tâche est une suite d'actions primitives à exécuter dans un ordre convenu pour résoudre un problème.

Les opérations primitives représentent les activités individuelles du bureau.



Entité TACHE

Cette entité a été définie dans le modèle organisationnel.

Entité OPERATION-PRIMITIVE

a. Définition.

Opération primitive sur un objet.

b. Attribut.

- NOM-OP-PRIM.
  Définition:
  C'est le nom de l'opération primitive.
  Propriétés:
  C'est un attribut simple,élémentaire et obligatoire.

Association COMPOSITION-TACHE.

Description	Une tâche se compose d'une suite d'actions primitives à exécuter dans un ordre convenu.
Liaison entre	TACHE et OPERATION-PRIMITIVE.
Rôle	"est divisée en" pour TACHE "fait partie de" pour OPERATION-PRIMITIVE.
Connectivité	1-N pour TACHE 0-N pour OPERATION-PRIMITIVE.

3.2.6.2 FORMALISATION DES OPERATION PRIMITIVES

L'analyse des besoins du bureau permet de faire un relevé des opérations primitives.

Pour chacune de ces opérations, nous donnerons:

- son effet : C'est le résultat de l'exécution de l'opération.
- sa formalisation.

A. LA SYNTAXE UTILISEE

Pour décrire les opérations primitives, nous utiliserons la syntaxe suivante:

- Les mots en majuscule sont des mots réservés.
- Les mots en minuscule sont des noms dont les valeurs sont choisies par l'utilisateur.
- Les mots entourés de crochets [ ] sont des clauses optionnelles à l'intérieur d'une construction.
- Les mots entourés de parenthèses ( ) sont des clauses répétitives.
- Les mots entourés d'accolades { } et disposés en colonnes ne peuvent être utilisés simultanément mais au moins un des mots doit être utilisé.

B. LES OPERATIONS PRIMITIVES

* CREER UN OBJET

Le travailleur de bureau crée un nouvel objet.

```
CREER
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
  { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
  { FICHIER  nom-fichier }
  { PILE  nom-pile }
```

* REPRODUIRE UN OBJET

Il s'agit de produire une ou plusieurs copies d'un objet (message, formulaire ou document)

```
COPIER  [ n  FOIS ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
```

Remarque: Si n est omis, on copie l'objet une fois.

* RECEVOIR UN OBJET

On reçoit un objet (message, formulaire, document ou dossier) d'un tiers, soit une personne ou une unité organisationnelle, de la même organisation ou non que le récepteur.

```
RECEVOIR  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
  { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
```

* COMMUNIQUER UN OBJET A UN OU PLUSIEURS DESTINATAIRES

Il s'agit de transmettre un objet (message, formulaire, document ou dossier) à un autre endroit. Le(s) destinataire(s) pourrai(en)t être une personne ou une unité organisationnelle, de la même organisation ou non que l'expéditeur, ou une liste de personnes faisant partie d'un même projet.

```
COMMUNIQUER  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
  { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
A  { PERSONNE  nom-personne  MEMBRE-UO  nom-uo }
   { UNITE ORGANISATIONNELLE  nom-uo }
   { PERSONNES DU PROJET  nom-projet }
[ ( , A  { PERSONNE  nom-personne  MEMBRE-UO  nom-uo }
         { UNITE ORGANISATIONNELLE  nom-uo }
         { PERSONNES DU PROJET  nom-projet }          ) ]
[ ( EN REPONSE A  [ COPIE NUMERO  numero-copie DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form } ) ]
```

* CONSULTER UN OBJET

Il s'agit de prendre connaissance du contenu de l'objet (le contenu d'un message, le corps d'un document, un ou plusieurs éléments variables d'un formulaire). Après consultation, l'objet est remis où il se trouvait.

a. consultation d'un message

```
CONSULTER MESSAGE nom-message IDENTIFIE PAR nom-message
```

b. consultation d'un formulaire

```
CONSULTER
 { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
 { PARTIE nom-partie DU FORMULAIRE titre-form
                          IDENTIFIE PAR  identifiant-form }
 { ELT-VARIABLE nom-elt-variable DE LA PARTIE nom-partie
   DU FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
```

c. consultation d'un document: le document pourra être consulté sur base de son identifiant ou par recherche selon un mot-clé.

```
CONSULTER
 { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
 { [ n ]  DOCUMENT TEL QUE SUJET-DOC CONTIENT nom-mot-clé
                                              [(,nom-mot-clé)]
}
```

Remarque: n = nombre de documents que l'on désire consulter.
Si n est omis, on les donne tous.

d. consultation d'un dossier: on peut consulter tous les constituants du dossier ou un seul constituant sur base de son identifiant.

```
FEUILLETER DOSSIER DE TYPE type-dossier
                   IDENTIFIE PAR identifiant-dossier

CONSULTER  [ COPIE NUMERO  numero-copie  DE ]
 { MESSAGE nom-message IDENTIFIE PAR nom-message }
 { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
 { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
DANS DOSSIER DE TYPE type-doss
        IDENTIFIE PAR identifiant-doss
```

e. consultation d'un fichier: on peut consulter un ou tous les constituants du fichier.

```
FEUILLETER  FICHIER  nom-fichier

CONSULTER  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
 DANS  FICHIER  nom-fichier
```

f. consultation d'une pile: on peut consulter un ou tous les constituants d'une pile.

```
FEUILLETER  PILE  nom-pile

CONSULTER  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
 DANS  PILE  nom-pile
```

* DETRUIRE UN OBJET

Il s'agit de détruire un objet (message, formulaire ou document) de façon permanente.

```
DETRUIRE  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
```

* VERIFIER UN OBJET

Vérification d'un formulaire: On vérifie qu'un ou plusieurs éléments d'un formulaire sont remplis et sont conformes à leur format.

Vérification d'un dossier: On vérifie que tous les constituants qui doivent figurer dans le dossier sont présents.

```
VERIFIER COMPLETUDE DOSSIER DE TYPE type-dossier
                      IDENTIFIE PAR identifiant-dossier

VERIFIER
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { PARTIE nom-partie DU FORMULAIRE titre-form
                          IDENTIFIE PAR identifiant-form }
  { ELT-VARIABLE nom-elt-variable DE LA PARTIE nom-partie
    DU FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
```

* MODIFIER UN DOCUMENT

Il s'agit de modifier le corps d'un document. Cette opération est utilisée lorsqu'une partie de texte doit être changée. Cette correction se fait à partir d'un brouillon.

```
MODIFIER DOCUMENT DE TYPE type-document
         IDENTIFIE PAR identifiant-document
  AU MOYEN DE MESSAGE BROUILLON nom-message
              IDENTIFIE PAR  nom-message
```

* ARCHIVER UN OBJET

Il s'agit d'archiver un objet dans une unité organisationnelle.

```
ARCHIVER
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
  { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
  { FICHIER nom-ficher }
  DANS  UNITE ORGANISATIONNELLE  nom-uo
```

* COMPLETER UN FORMULAIRE

Il s'agit de donner une valeur à un ou plusieurs éléments variables d'un formulaire. On pourra compléter tout le formulaire, une partie du formulaire ou un élément du formulaire.

```
COMPLETER
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { PARTIE nom-partie DU FORMULAIRE titre-form
                              IDENTIFIE PAR identifiant-form }
  { ELT-VARIABLE nom-elt-variable DE LA PARTIE nom-partie
  DU FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
```

* CLASSER UN OBJET ELEMENTAIRE DANS UN OBJET STRUCTURANT

Il s'agit de mettre dans un objet structurant (dossier, fichier ou pile) un objet élémentaire (message, formulaire ou document).

```
CLASSER  [ COPIE NUMERO  numero-copie  DE ]
  { MESSAGE nom-message IDENTIFIE PAR nom-message }
  { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
  { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
 DANS { DOSSIER DE TYPE type-doss
             IDENTIFIE PAR identifiant-doss }
      { FICHIER  nom-fichier }
      { PILE  nom-pile }
```

* ENLEVER UN OBJET ELEMENTAIRE D'UN OBJET STRUCTURANT

Il s'agit de retirer un objet élémentaire d'un objet structurant. L'objet enlevé ne sera pas détruit; il pourra être reclassé ailleurs après modification. On peut accéder à l'objet à enlever sur base de son identifiant ou par un accès LIFO.

```
ENLEVER  [ COPIE NUMERO  numero-copie  DE ]
   { MESSAGE nom-message IDENTIFIE PAR nom-message }
   { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
   { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifiant-doc }
 DE { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
    { FICHIER  nom-fichier }
    { PILE nom-pile }

ENLEVER  PREMIER CONSTITUANT  DE PILE  nom-pile
```

* PRENDRE UNE DECISION

Une personne légalement habilitée à le faire va prendre une décision concernant l'acceptation, le rejet, la mise en attente,... d'une proposition. Cette décision sera prise à partir d'un dossier, d'un document, d'un formulaire.

```
DECIDER  A PARTIR DE
   { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifant-doc }
   { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
   { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
```

* NEGOCIER

Cette opération a le même effet que la prise de décision mais la personne qui devrait décider n'a pas l'information ni/ou l'autorité pour prendre la décision elle-même et doit donc faire appel à d'autres personnes pour décider avec elle.

```
NEGOCIER  A PARTIR DE
   { DOCUMENT DE TYPE type-doc IDENTIFIE PAR identifant-doc }
   { FORMULAIRE titre-form IDENTIFIE PAR identifiant-form }
   { DOSSIER DE TYPE type-doss IDENTIFIE PAR identifiant-doss }
 AVEC  PERSONNE  nom-personne
    [ ( , AVEC  PERSONNE  nom-personne ) ]
```

3.2.7 LE MODELE DE LA DYNAMIQUE DES TRAITEMENTS

On distinguera deux types de dynamique:

- La dynamique globale:
  Elle permet de décrire l'enchainement des tâches pour réaliser un certain objectif.

- La dynamique interne à une tâche:
  Elle permet de décrire l'enchainement des opérations primitives à l'intérieur d'une tâche.

3.2.7.1 LA DYNAMIQUE GLOBALE

Plusieurs choix sont possibles pour décrire l'enchainement des tâches. Nous donnerons deux approches différentes:

a. Une approche où les tâches sont déclenchées par des événements: c'est l'approche étudiée dans DSL.

b. Une approche basée sur la définition de préconditions et de postconditions.

A. L'APPROCHE DE DSL

Nous en rappellerons les concepts de base. On pourra trouver plus de détails dans [BOD].

La dynamique de DSL repose sur deux concepts de base: le processus et l'événement.

Un processus est l'exécution d'une procédure de traitement de l'information dont la progression peut être représentée par son état. Le cycle de vie d'un processus comporte les états: déclenché, actif, interrompu et terminé. La notion de changement d'état en découle; on a les changements: activation, interruption, redémarrage et terminaison.

Un événement correspond à un changement d'état du système caractérisé dans le temps et dans l'espace. On distingue des événements internes et externes.

Les spécifications dynamiques décrivent les enchainements entre les traitements. Les enchainements possibles sont:

- L'enchainement séquentiel:
  Un événement provoque le déclenchement d'un processus.

- L'enchainement éclaté:
  Un événement provoque le déclenchement simultané de plusieurs processus.

- L'enchainement multiple:
  Un événement provoque le déclenchement de plusieurs occurrences d'un même processus.

- L'enchainement conditionnel:
  Un événement déclenche un processus A si une condition spécifiée est vraie ou déclenche un processus B sinon.

- L'enchainement convergent:
  Le déclenchement d'un processus est provoqué par un événement EV1 ou EV2 ou ... ou EVn.

- L'enchainement synchronisé:
  Un processus n'est déclenché que lorsque les événements EV1 et EV2 ont lieu. On impose une certaine attente avant que les événements ne produisent leurs effets dynamiques.

B. L'APPROCHE PAR PRECONDITIONS ET POSTCONDITIONS

Cette approche contient une description de l'état de l'environnement requis pour que la tâche commence: ce sont les préconditions.

Après la terminaison de la tâche, certaines conditions doivent être satisfaites: ce sont les postconditions.

Définitions:

La précondition est une condition qui doit expliciter les propriétés des arguments (objets) qui doivent être vérifiées avant toute exécution de la tâche pour que celle-ci s'exécute correctement. Ces propriétés porteront sur la valeur de l'état des objets donnés en arguments et éventuellement, sur la valeur de certains autres attributs.

La postcondition est une condition qui doit expliciter les propriétés des résultats de la tâche, qui doivent être satisfaites à la fin de toute exécution de la tâche si celle-ci s'est exécutée correctement.

Les postconditions de certaines tâches correspondent aux préconditions d'autres tâches, ce qui permet un enchainement dynamique.

Des exemples illustrant cette approche par préconditions et postconditions sont décrits au chapitre 4 (3.3. Description de l'exemple de base avec le MIB; 3.3.2 Description des tâches).

3.2.7.2 LA DYNAMIQUE INTERNE

La dynamique interne à une tâche, entre opérations primitives, reprend les mêmes structures dynamiques que celles que l'on peut rencontrer dans les langages de programmation évolués; à savoir:

a. La séquence.

```
opération primitive 1 ;
opération primitive 2.
```

b. La condition.

```
SI  condition
 ALORS  ( opération primitive )
 SINON  ( opération primitive )
```

c. L'itération.

On itère tant qu'une certaine condition est spécifiée ou on itère sur une séquence d'objets.

```
TANT QUE  condition  FAIRE  ( opération primitive )

POUR CHAQUE  objet  FAIRE  (opération primitive )
```

d. L'attente.

```
ATTENDRE JUSQU'A  condition
```

3.2.8 COHERENCE ET COMPLETUDE

Après avoir spécifié le comportement d'un système d'information de bureau à l'aide du modèle décrit, encore faut-il vérifier qu'il s'agit d'une bonne spécification de la solution choisie en s'assurant de l'utilisation correcte du modèle.

Une bonne spécification est une spécification:

- complète : il n'existe pas de composants référencés mais non complètement décrits;
- cohérente : il n'existe pas de contradiction dans la description.

3.2.8.1 REGLES DE COMPLETUDE

Une spécification sera dite incomplète si par rapport au modèle de spécification, des relations qui devraient caractériser un objet ne figurent pas dans la description des données.

A. COMPLETUDE DU SCHEMA CONCEPTUEL

a. Toute la réalité est bien représentée:
   - tous les objets sont représentés;
   - tous les traitements sont représentés.

b. Chacun des concepts devra être décrit complètement comme le précise le modèle. Pour chaque concept, on donnera les attributs demandés et les associations nécessaires, en respectant les contraintes d'intégrité.
   Pour chacun des objets informationnels, l'utilisateur n'oubliera pas de décrire complètement les états possibles. On devra en outre indiquer quels sont les états terminaux.

c. Tout objet informationnel est créé ou manipulé par une tâche.

   Tout objet n'ayant aucun rôle dans le schéma conceptuel n'en a pas non plus dans le système réel et figure dans le système informationnel à titre documentaire.

d. Toute ressource (ressource consommable, technologie ou personne) est requise par au moins une tâche.

e. Toute classe possède une propriété permanente: sa propriété identifiante.

f. Toutes les connectivités des relations, ainsi que leur rôle, sont décrites.

B. COMPLETUDE DU SCHEMA DE LA DYNAMIQUE

Pour toutes les classes de tâches citées dans le schéma:

a. Leur procédure est décrite, au moyen de la décomposition en opérations primitives et de la dynamique interne.

b. Dans un cycle dynamique, toute exécution d'une tâche entrainera des transformations d'information donnant lieu à une autre tâche, sauf en ce qui concerne la dernière du flux.
Dans l'approche par préconditions et postconditions, cette dernière tâche est telle que tous les objets dans la postcondition sont dans un état terminal.
Ainsi, toute tâche exprimera une action dynamique, sauf la dernière du flux.
Dans l'approche par préconditions et postconditions, cette action dynamique sera exprimée par le changement d'état d'un objet.

c. Le déclenchement de chaque tâche doit être prévu. Selon l'approche, on doit décrire les préconditions ou les événements déclencheurs de la tâche.

En ce qui concerne le déclenchement des tâches:

Si on prend l'approche de DSL:

a. Les attributs des événements sur lesquels porte une modalité de déclenchement sont spécifiés.

b. Il n'existe pas d'événement qui ne soit causé par au moins une tâche (événement interne) ou dont l'échéancier n'ait pas été établi.

c. Il n'existe pas d'événement qui ne déclencherait aucune tâche, s'il ne s'agit d'événement terminal.

Si on prend l'approche par préconditions et postconditions:

a. Pour chaque tâche, on a défini sa précondition et sa postcondition.

b. Il n'existe pas de préconditions qui ne soient jamais vérifiées.

c. Il n'existe pas de postcondition qui ne déclencherait aucune tâche, s'il ne s'agit pas d'une postcondition terminale, c'est-à-dire, il n'existe pas de postcondition ne correspondant à la précondition d'aucune tâche et telle que tous les états des objets y apparaissant ne sont pas des états terminaux.

3.2.8.2 REGLES DE COHERENCE

Une spécification sera dite non cohérente si elle présente des contradictions ou des incompatibilités.

A. COHERENCE DU SCHEMA CONCEPTUEL

a. Il ne devra pas y avoir de contradictions syntaxiques.
Exemple: Si un objet a été déclaré et stocké antérieurement comme un DOCUMENT, il ne pourra plus être décrit comme MESSAGE.

b. En ce qui concerne la sémantique:
Des valeurs d'un attribut prédéfini devront appartenir à un domaine donné si ce domaine a été spécifié dans la description formelle.

c. Absence d'homonymes:
Les objets de la description doivent posséder un nom unique dans toute la description.

d. La définition de tout élément d'une structure d'information doit être vraie pour toute la durée de vie du système d'information.

e. Les contraintes d'intégrité données dans le modèle sont toutes respectées.

B. COHERENCE DU SCHEMA DE COMPORTEMENT

a. Les modalités de déclenchement et de survenance ne sont pas contradictoires entre elles.

Approche DSL: Il y a contradiction lorsqu'une situation de décision empêche systématiquement une situation de synchronisation de se produire car certains événements contribuant au déclenchement d'une même tâche ne peuvent jamais être simultanément présents.

Approche par préconditions et postconditions: Une précondition pourrait ne jamais être vérifiée car certaines propriétés de cette précondition contribuant au déclenchement d'une tâche ne peuvent jamais être vérifiées simultanément.

b. Dans l'approche par préconditions et postconditions, les préconditions et les postconditions doivent être cohérentes avec les états des objets décrits par l'utilisateur. C'est-à-dire, les états repris dans les préconditions et les postconditions sont des états possibles de ces objets.

c. Tout état décrit dans la description des objets devra être atteint à un moment, c'est-à-dire, apparaitra dans les postconditions d'une tâche ou dans la description d'un événement de déclenchement. Il n'existe pas de tâche qui ne soit jamais déclenchée parce qu'un état ne serait jamais atteint.

d. Les propriétés des réquisitions de ressource par les différentes tâches ne sont pas contradictoires avec le calendrier de disponibilité des ressources requises.
Exemples d'incohérence:
- un type de tâche T requiert différentes ressources dont l'intersection des calendriers est vide.
- une tâche est non interruptible et sa durée estimée d'exécution est supérieure à la période la plus large du calendrier de disponibilité des ressources requises.
- un taux de réquisition est supérieur à la capacité maximum de la ressource requise.

e. Le graphe de la dynamique ne possède pas de circuit ne comportant aucune entrée et / ou aucune sortie.

f. Il n'existe pas dans le fonctionnement du système des circuits infinis.

g. Tout circuit dans le fonctionnement du système correspond à une situation réelle.

h. Toute situation finale dans le fonctionnement du système correspond à une situation réelle.

Si des modifications sont apportées au système d'information de bureau, celles-ci devront lui conserver ses qualités de fidélité par rapport à la réalité, de complétude et de cohérence. Après toute modification, le système d'information devra donc toujours vérifier les règles énoncées ci-avant.

CHAPITRE 4: IMPLEMENTATION DU MIB EN DSL

1 INTRODUCTION

Pour implémenter le modèle décrit au chapitre 3, deux solutions se présentent:

a. Ecrire complètement un système logiciel pour implémenter le modèle. Ce système logiciel comprendrait:
   - le langage de spécification adapté au modèle de bureau;
   - des outils documentaires;
   - des outils de contrôle de cohérence;
   - des outils de simulation, c'est-à-dire d'évaluation du caractère réalisable des spécifications du système d'information de bureau.

b. Implémenter notre modèle détaillé à partir d'un système plus générique.

Nous avons choisi cette deuxième solution. Le système utilisé est le système IDA (Interactif Design Approach). On peut trouver une description de ce système dans [BOD], [IDA 1], [IDA 2] et [IDA 3].

2 JUSTIFICATION DU CHOIX DE IDA

Nous avons choisi le système IDA pour trois raisons:

Première raison:

Le système IDA existe. Cela permet donc de récupérer toute la technologie et les outils existants plutôt que de créer un système complet.

Deuxième raison:

Le système IDA fournit plusieurs résultats qui correspondent aux objectifs que nous nous sommes fixés. Ces objectifs, énoncés dans l'introduction, concernaient:
- la spécification;
- la documentation;
- la simulation.

En effet, le système-logiciel IDA se décrit comme suit:

- Une base de données des spécifications regroupe l'ensemble des spécifications du système d'information introduites et devient ainsi la source unique de toute la documentation.
- Les spécifications sont décrites à l'aide du langage DSL (Dynamic Specification Language) et sont intégrées dans la base de données par des programmes de mise-à-jour faisant partie de l'analyseur DSA (Dynamic Specification Analyser).
- L'analyseur comprend en outre un ensemble de rapports documentaires et d'analyse présentant sous des formes variées (narratives, listes, tableaux, graphiques) différents aspects du système spécifié, à l'usage des diverses personnes impliquées dans le projet: responsables, analystes, programmeurs et utilisateurs.
- Il existe un langage interactif d'interrogation de la base de données (Query System) qui a pour objectif de trouver des objets de la base qui vérifient des critères de sélection.
  L'emploi du langage d'interrogation peut être du type documentaire (production de listes d'objets vérifiant certains critères).
  On peut également utiliser le langage d'interrogation comme moyen d'expression de contraintes d'intégrité en vue de contrôler certains aspects de la cohérence de la base de données des spécifications (vérification qu'une liste d'objets satisfait à certains critères).
- Il existe également un interface vers un générateur automatique d'un programme de simulation pour évaluer le caractère réalisable du système décrit.

Le langage DSL est un langage non procédural basé sur le modèle Entité-Association. En effet, il est construit à partir des notions suivantes:

- des types d'OBJET dont le nombre est fixé dans le langage;
- des types de RELATION entre ces objets, dont le nombre est également limité;
- des PROPRIETES associées à ces types d'objet ou de relation.

Notre modèle étant également fondé sur ce modèle Entité-Association, nous espèrons pouvoir le décrire à l'aide du langage DSL. Il faudra cependant voir si les limitations de DSL en ce qui concerne les objets et les relations ne seront pas trop restrictives pour permettre une description complète de notre modèle. Dans ce cas, le langage devrait être étendu pour représenter tous les concepts du MIB.

Parmi les rapports documentaires fournis par l'analyseur DSA, certains correspondent à la description de différents aspects mentionnés dans l'introduction. Ce sont notamment les rapports:

- FORMATTED-STATEMENT(1) et SELECTIVE-FORMATTED-STATEMENT(2)
  Ces rapports restituent la totalité (1) ou un sous-ensemble (2) des spécifications relatives à un objet donné. Ce rapport correspond donc à la fonction voulue d'extraction de certaines descriptions à partir des spécifications décrites: étant donné un objet de bureau, on pourrait visualiser toute l'information dont on dispose.
  Exemple:
  Si on fournit le nom d'un message, un tel rapport fournirait les renseignements suivants:
  Par une analyse de la description du message dans le modèle de structuration des informations, on obtiendrait:
  - le type du message;
  - son contenu;
  - ses états possibles;
  - son statut (original ou copie);
  - son support

  ainsi que des renseignements sur son classement éventuel dans un objet structurant.
  Se basant sur les relations avec le modèle de structuration de l'organisation, le rapport donnerait:
  - l'origine du message;
  - les destinations successives du message.

  Enfin, par extraction dans le modèle de la structuration des traitements, le rapport décrirait les tâches du bureau qui manipulent ce message.

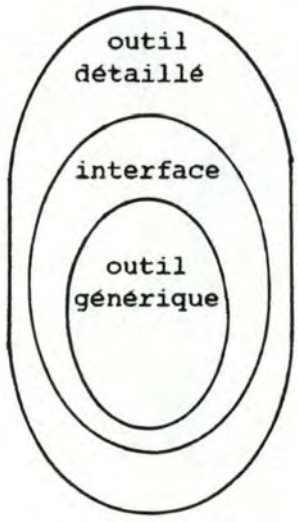
- EXTENDED-PICTURE(1) ET STRUCTURE(2)
  Ces rapports présentent sous forme graphique (1) ou sous forme d'une liste décalée (2), pour une liste de noms d'objets donnés en entrée, une hiérarchie de noms reliés aux précédents par un nom de relation.
  Ce rapport peut nous permettre de tracer un diagramme de flux.
  - diagramme de flux dans l'organisation:
    Si on fournit le nom de l'objet et qu'on spécifie les relations DEST-OBJET-UO, DEST-OBJET-PERS, EXPED-OBJET-UO et EXPED-OBJET-PERS, qui relient l'objet à l'organisation, on obtiendra la liste de tous les destinataires et de tous les expéditeurs de l'objet, c'est-à-dire le flux de l'objet dans l'organisation.
  - diagramme de flux parmi les tâches:
    Si on fournit le nom d'un objet et que l'on spécifie les relations DONNEES et RESULTATS qui décrivent le lien entre l'objet et les tâches qui le manipulent, on obtiendra la liste de toutes les tâches successives qui concernent l'objet.

- FUNCTION-FLOW-DATA
  Ce rapport rappelle l'inventaire des informations en entrée et en sortie d'un ou plusieurs traitements. Ainsi, étant donné le nom d'une tâche de bureau, on pourrait obtenir la liste de tous les objets de bureau qu'elle consulte, modifie et / ou crée.

- PROCESS-REQUEST-RESSOURCE
  Ce rapport présente, sous forme d'un tableau de synthèse, les relations de réquisition des ressources par les traitements. Il est utile pour l'analyse de ces relations préalablement au lancement d'une simulation.

Troisième raison:

La troisième raison de choisir le système IDA correspond à une philosophie de base consistant à utiliser un outil commun générique sur lequel on grefferait différents interfaces de façon à implémenter divers systèmes plus détaillés. Nous voulons décrire ici un système adapté au bureau; on pourrait en imaginer d'autres, par exemple, un modèle adapté au travail de l'architecte, ... Tous les aspects particuliers du système détaillé que l'on élaborerait seraient rejetés dans l'interface.

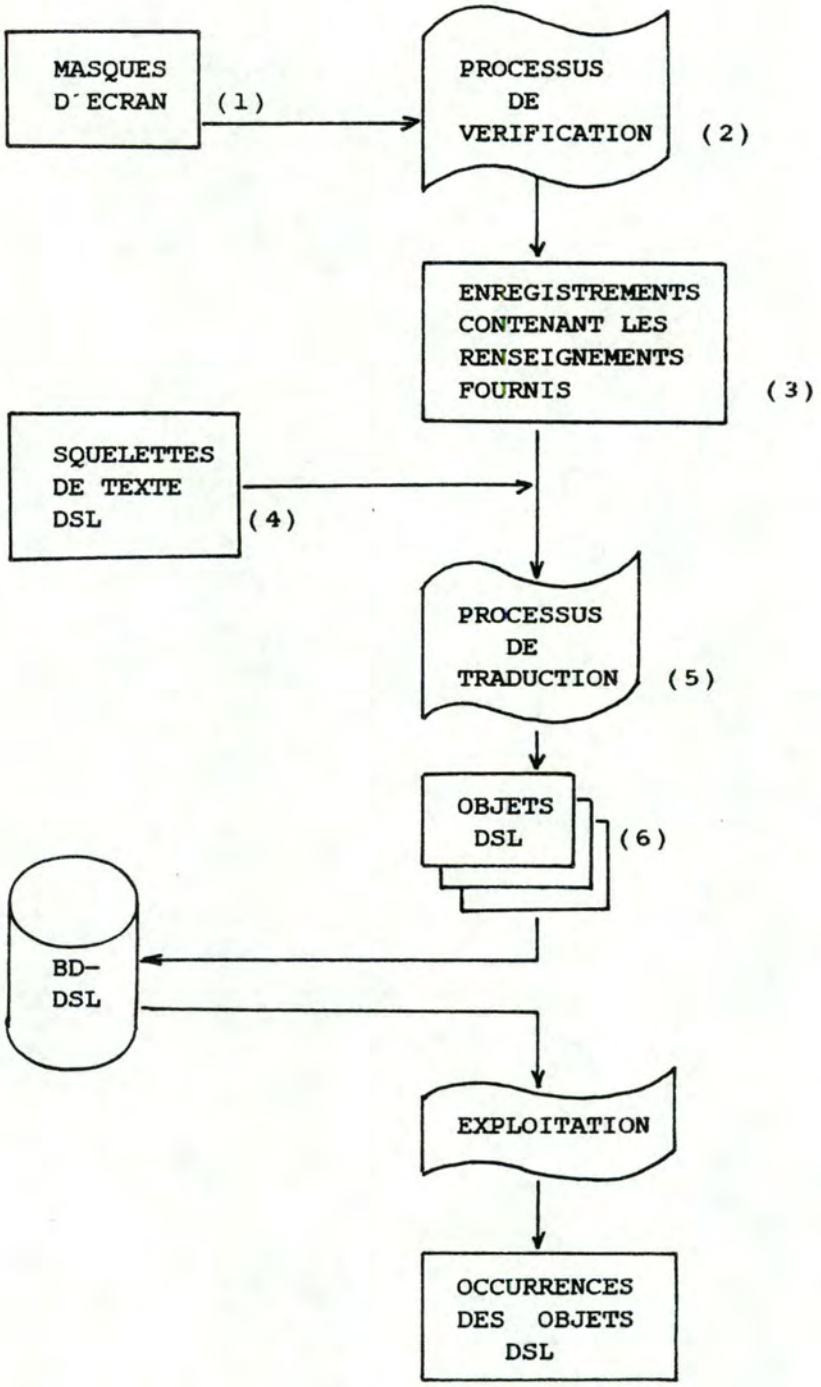


3 LA REPRESENTATION DES OBJETS INFORMATIONNELS

Dans le cadre de ce mémoire, nous nous limiterons à la représentation des objets informationnels du bureau en DSL. Cette représentation devra nous permettre d'atteindre les objectifs proposés: aide à la conception, documentation et simulation. Cependant, comme nous nous intéressons uniquement à la représentation des objets informationnels et non aux tâches, nous aborderons peu la dynamique du modèle, et donc la simulation.

3.1 SCHEMA GENERAL DE TRANSFORMATION DES OBJETS INFORMATIONNELS

Le passage de la description des objets selon le modèle de structuration des informations décrit au chapitre 3 à une description des objets dans le langage DSL - description exploitable à des fins de documentation et de simulation - nécessite plusieurs étapes de transformation que l'on peut résumer schématiquement comme suit:



(1) L'utilisateur décrit les objets du bureau selon le modèle de structuration des informations.
Afin de faciliter cette description, on peut lui fournir des écrans de saisie; il lui restera alors à donner les renseignements demandés.
On peut l'aider davantage en lui rappelant, en plus des concepts requis:

- le format de chacun de ces concepts;
  Exemple: le nom du message est alphabétique; il est composé de x caractères au maximum.
- les règles de cohérence et de complétude se rapportant à l'objet décrit.
  Exemples:
  - deux objets ne peuvent avoir le même nom;
  - pour chaque objet, on doit décrire au moins un état;
  - ...

L'objectif de cette première étape du schéma de transformation est la recherche d'un interface agréable , qui aiderait le concepteur dans la spécification de son système bureautique.

(2) Processus de vérification.
Il s'agit de voir si les descriptions fournies par l'utilisateur respectent bien toutes les règles de cohérence et de complétude définies pour les objets du modèle (cf. chapitre 3).

(3) Enregistrements.
L'information introduite par l'utilisateur et vérifiée sera rangée dans des enregistrements prédéfinis.

(4) Squelettes DSL.
Pour chacun de ces objets, il faudra définir des squelettes de texte DSL, constituant un ensemble de DSL utile à la traduction de nos objets.

(5) Ces squelettes seront complétés avec les informations collectées dans les enregistrements lors du processus de traduction.

(6) Après traduction, on aura obtenu la représentation des objets de notre modèle en DSL. Ces représentations seront rangées dans une base de données DSL que l'on pourra exploiter à des fins de documentation et simulation.

3.2 EXEMPLE DE BASE

Afin de faciliter la compréhension de la transformation des objets informationnels en DSL, nous baserons notre raisonnement sur un exemple simplifié du flux administratif lié à l'inscription des étudiants à l'institut. Nous ne considèrerons que les étudiants venant des candidatures en sciences économiques et sociales ou en mathématiques des F.N.D.P. et ayant pris contact avec le secrétariat, jusqu'à l'ouverture de leur dossier.

On peut trouver la description de cette application à l'annexe 1.

3.3 DESCRIPTION DE L'EXEMPLE DE BASE AVEC LE MIB

Décrivons maintenant le flux présenté dans l'exemple de base à l'aide du modèle de bureau proposé au chapitre 3. Nous procèderons pour cela en deux phases:

- description des objets;
- description des tâches et de leur enchainement.

3.3.1 DESCRIPTION DES OBJETS

Les concepts repris dans chacun des objets sont semblables. Nous nous attacherons donc à décrire complètement un seul objet élémentaire: le document et un seul objet structurant: le fichier. Nous citerons uniquement les autres objets, avec la liste de leurs états (ces renseignements nous suffirons par la suite).

3.3.1.1 DESCRIPTION DES DOCUMENTS

Pour chaque document, nous donnerons son nom, son type, la liste de ses états possibles en précisant les états terminaux, son expéditeur et ses destinations successives, les autres objets qu'il peut recevoir en réponse.

La définition complète de tous ces concepts a été donnée au chapitre 3.

- le document lettre-contact

  Pour s'inscrire, l'étudiant peut prendre contact avec le secrétariat en envoyant une lettre.

  NOM: lettre-contact;
  TYPE: lettre;
  ETATS POSSIBLES: reçu-secrétariat,
  traité,
  classé-fichier-contacts,
  classé-dossier-étudiant (terminal);
  EXPEDITEUR: étudiant;
  DESTINATAIRE: secrétariat-administratif;
  REPONSE: lettre-acceptation (formulaire),
  demande-inscription (formulaire).

3.3.1.2 DESCRIPTION DES MESSAGES

Nous donnerons seulement une description succincte des messages, avec leur nom et la liste de leurs états possibles.

- le message téléphone-contact

  L'étudiant peut prendre contact avec le secrétariat par téléphone.

  NOM: téléphone-contact;
  ETATS POSSIBLES: reçu-secrétariat,
  traité (terminal).

- le message entretien-contact

  L'étudiant peut prendre contact avec le secrétariat en s'y présentant.

  NOM: entretien-contact;
  ETATS POSSIBLES: reçu-secrétariat,
  traité (terminal).

- le message rentrée-académique

  Il indique un événement temporel.

  NOM: rentrée-académique;
  ETATS POSSIBLES: attente,
  survenu (terminal).

3.3.1.3 DESCRIPTION DES FORMULAIRES

Nous donnerons seulement une description succincte des formulaires, avec leur nom et la liste de leurs états possibles.

- le formulaire lettre-acceptation

  Quand l'étudiant a pris contact avec le secrétariat, la secrétaire lui envoie une lettre décrivant les modalités d'inscription.

  NOM: lettre-acceptation;
  ETATS POSSIBLES: envoyé-étudiant (terminal).

- le formulaire demande-inscription

  Pour s'inscrire, l'étudiant doit remplir un formulaire de demande d'inscription en deux exemplaires.

  NOM: demande-inscription;
  ETATS POSSIBLES: à-remplir,
  rempli,
  à-corriger,
  à-compléter,
  complet-et-correct,
  classé-fichier-inscriptions,
  classé-dossier-étudiant (terminal).

3.3.1.4 DESCRIPTION DES DOSSIERS

Pour chaque dossier, nous donnerons son nom et la liste de ses états possibles.

- le dossier réponse

  Lorsque l'étudiant a pris contact avec la secrétaire, elle lui envoie une réponse comprenant le formulaire lettre-acceptation et deux formulaires demande-inscription.

  NOM: réponse;
  ETATS POSSIBLES: envoyé-étudiant (terminal).

- le dossier étudiant

  A la rentrée académique, la secrétaire reprend les demandes d'inscription remplies et pour chaque étudiant inscrit, elle ouvre un dossier où elle range une demande d'inscription remplie et éventuellement la lettre de prise de contact envoyée par l'étudiant.

  NOM: étudiant;
  ETATS POSSIBLES: ouvert,
  complet (terminal).

3.3.1.5 DESCRIPTION DES FICHIERS

Pour chaque fichier, nous donnerons son nom, la liste de ses états possibles et le nom de l'objet constituant.

- le fichier contacts

  On y range les lettres de prise de contact envoyées par les étudiants.

  NOM: contacts;
  ETATS POSSIBLES: ouvert,
  vide (terminal);
  CONSTITUANT: lettre-contact.

- le fichier inscriptions

  On y range les demandes d'inscription remplies correctement et complètement par l'étudiant.

  NOM: inscriptions;
  ETATS POSSIBLES: ouvert,
  clôturé,
  vide (terminal);
  CONSTITUANT: demande-inscription.

3.3.2 DESCRIPTION DES TACHES

Nous allons maintenant décrire les tâches et leur enchainement. Nous utiliserons, pour cela, la dynamique par préconditions et postconditions.

- Tâche 1: traitement-contact-étudiant.

  La secrétaire reçoit un contact de l'étudiant (écrit ou oral). Elle prépare une réponse à envoyer, consistant en une lettre d'acceptation et une demande d'inscription en double exemplaire. Elle envoie la réponse.

  Préconditions
  (∃ MESSAGE téléphone-contact
  t.q. ETAT-MESSAGE = envoyé-secrétariat;

  ∧ ∃ MESSAGE entretien-contact
  t.q. ETAT-MESSAGE = reçu-secrétariat;

  ∧ ∃ DOCUMENT lettre-contact
  t.q. ETAT-DOCUMENT = reçu-secrétariat;)

  Postconditions
  (∃ DOSSIER réponse
  t.q. ETAT-DOSSIER = envoyé-étudiant;

  ∧ ∃ DOCUMENT lettre-contact
  t.q. ETAT-DOCUMENT = traité;)

- Tâche 2: classement-lettre-contact

  Si le contact de l'étudiant était un contact écrit, la secrétaire le classe dans le fichier des contacts.

  Préconditions
  (∃ DOCUMENT lettre-contact
  t.q. ETAT-DOCUMENT = traité;

  ∧ ∃ FICHIER contacts
  t.q. ETAT-FICHIER = ouvert;)

  Postconditions
  (∃ FICHIER contacts'
  t.q. contacts' = contacts +
  DOCUMENT lettre-contact
  t.q. ETAT-DOCUMENT =
  classé-fichier-contacts)

- Tâche 3: vérification-demande-inscription

La secrétaire vérifie si le formulaire de demande d'inscription est bien rempli. Si non, elle le renvoie à l'étudiant pour qu'il le corrige ou le complète.

Préconditions

```
( ∃  FORMULAIRE  demande-inscription
       t.q.  ETAT-FORMULAIRE = rempli;    )
```

Postconditions

```
(  ∃  FORMULAIRE  demande-inscription
       t.q.  ETAT-FORMULAIRE = complet-et-correct
                               à-corriger
                               à-compléter;    )
```

- Tâche 4: classement-demande-inscription

Quand la demande d'inscription est remplie correctement, la secrétaire la classe dans le fichier des inscriptions.

Préconditions

```
( ∃  FORMULAIRE  demande-inscription
       t.q.  ETAT-FORMULAIRE = complet-et-correct;

  ∧ ∃  FICHIER  inscriptions
       t.q.  ETAT-FICHIER = ouvert;        )
```

Postconditions

```
(  ∃  FICHIER  inscriptions'
       t.q.  inscriptions' = inscriptions +
               FORMULAIRE  demande-inscription
                 t.q.  ETAT-FORMULAIRE =
                         classé-fichier-inscriptions  )
```

- Tache 5: ouverture-dossier

A la rentrée académique, la secrétaire reprend les formulaires de demande d'inscription (rangés dans le fichier inscriptions) ainsi que les lettres de prise de contact (rangées dans le fichier contacts) et ouvre un dossier pour l'étudiant.

Préconditions

(∃ MESSAGE CALENDRIER rentrée-académique
t.q. ETAT-MESSAGE = survenu;

∧ ∃ FICHIER inscriptions
t.q. ETAT-FICHIER = ouvert;

∧ ∃ FICHIER contacts
t.q. ETAT-FICHIER = ouvert;)

Postconditions

(∀ étudiant t.q.
∃ FORMULAIRE demande-inscription
t.q. ETAT-FORMULAIRE
= classé-fichier-inscriptions:
∃ DOSSIER etudiant
t.q. ETAT-DOSSIER = ouvert;)

3.4 PROPOSITION DE DESCRIPTION A L'AIDE DE DSL

3.4.1 LES OBJETS DONT ON DISPOSE EN DSL

Avant de décrire les objets du modèle proposé, nous allons rappeler brièvement les objets présents dans le langage DSL.

On se reportera pour des explications complémentaires à [BOD].

A. Les objets du langage DSL permettant l'expression de la statique des informations et des traitements.

On peut trouver une définition de ces concepts dans le langage DSL à l'annexe 2.

- l'entité: ENTITY
  L'entité permet de représenter une chose concrète ou abstraite à propos de laquelle on veut enregistrer des informations.

- le message: MESSAGE
  Le message véhicule des informations définies dans la mémoire du système d'information.

- la relation: RELATION
  Une relation est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné.

- les propriétés d'objet: ELEMENT ou GROUP
  Caractéristique ou qualité d'une entité, d'un message ou d'une relation.

- le processus: PROCESS
  Un processus est l'exécution d'une procédure de traitement de l'information. C'est le concept parallèle à notre concept de tâche.

- l'interface: INTERFACE
  L'objet INTERFACE permet de décrire sommairement l'organisation.

- l'ensemble: SET
  Un SET est une collection d'entités et / ou messages et / ou relations.

- l'attribut: ATTRIBUTE
  Un attribut permet d'associer une caractéristique, non initialement prévue dans la définition du langage, et la valeur qu'elle prend pour l'objet considéré.

- le mot-clé: KEYWORD
  L'objet KEYWORD permet d'associer des mots-clé à un objet. Ces mots-clé sont notamment utilisés comme critères de sélection des objets appartenant à la base de données des spécifications.

B. Les concepts du langage DSL permettant l'expression de la dynamique.

Nous avons déjà rappelé ces différents concepts lorsque nous avons parlé de la dynamique globale du modèle proposé. Pour rappel, il y avait trois concepts importants: les processus, les événements et les structures d'enchainement.

3.4.2 LA TRANSFORMATION DES OBJETS INFORMATIONNELS EN DSL

Trois concepts apparaissent dans la description des objets informationnels du MIB:
- les objets informationnels proprement dits;
- des liens entre les objets informationnels, traduisant la notion de réponse attendue;
- des liens avec l'organisation et son environnement, traduisant l'origine et les destinations successives des objets.

Pour la transformation des objets informationnels en DSL, on peut adopter une démarche calquée sur ces trois concepts:

- Nous allons d'abord étudier la transformation des objets informationnels proprement dits, du modèle proposé en DSL.

- Nous verrons ensuite comment transformer le concept de réponse attendue.

- Enfin, on tentera de résoudre les problèmes liés à la transformation des liens avec l'organisation.

3.4.2.1 LA TRANSFORMATION DES OBJETS INFORMATIONNELS PROPREMENT DITS

Nous procèderons à une description de ces objets à deux niveaux:

- un niveau statique, qui nous permettra de remplir l'objectif de documentation;
- un niveau dynamique, qui nous permettra de remplir l'objectif de simulation du modèle de bureau décrit.

La dynamique ne sera envisagée ici que très partiellement car cela ne fait pas l'objet de l'étude restreinte actuelle. Nous l'aborderons seulement pour analyser son influence sur la description des objets informationnels. Nous abandonnerons, dans un premier temps, les notions de préconditions et postconditions retenues dans le modèle de bureau. Nous décrirons la dynamique comme en DSL, en nous basant sur les concepts de processus et d'événement. Nous conserverons toutefois l'idée d'états successifs des objets informationnels, pour l'implantation ultérieure du modèle.

A. LE NIVEAU STATIQUE

La première étape de la recherche d'une solution possible de transformation des objets informationnels en DSL consiste à étudier comment passer d'une description d'un objet dans le MIB à sa description en DSL. Nous allons donc donner d'abord une vue purement statique de la transformation. A ce niveau statique de transformation, nous verrons:

- comment transformer les objets pris de façon isolée.

  La structure des objets étant semblable, nous décrirons uniquement les objets informationnels document et fichier. La description de la transformation des autres objets informationnels se trouve à l'annexe 3.

- comment insérer ces objets dans un diagramme de flux de façon à décrire leur origine et leurs destinations dans l'organisation.

A.1. LA DESCRIPTION DES OBJETS PRIS ISOLEMENT

La solution de transformation qui semble la plus naturelle consisterait à transformer un objet du MIB en un seul objet DSL.

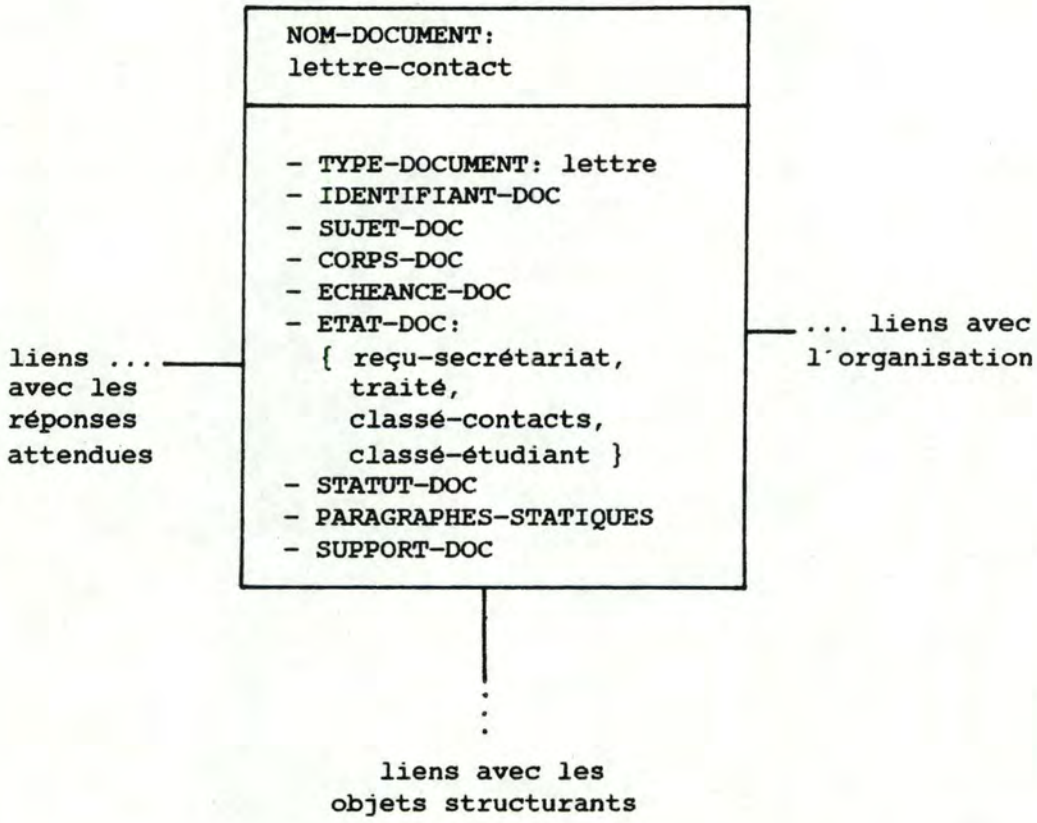
Voyons si cette solution suffirait à décrire complètement l'information contenue dans l'objet informationnel du MIB.

A.1.1. DESCRIPTION D'UN OBJET ELEMENTAIRE: LE DOCUMENT

Prenons à titre d'exemple le document lettre-contact.

* LE DOCUMENT DANS LE MODELE PROPOSE

Dans le modèle proposé, on avait:



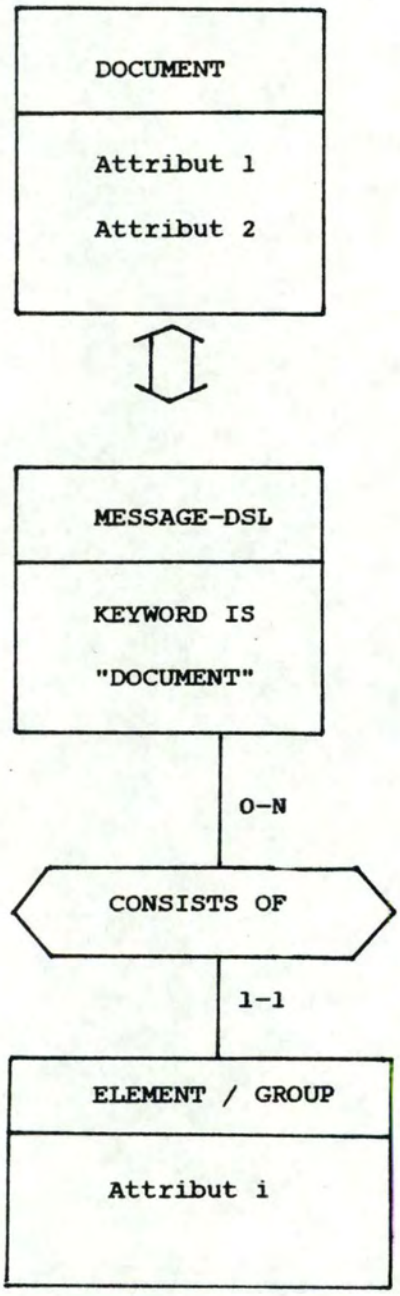
* LE DOCUMENT EN DSL

Le document dans le modèle proposé
- définit une partie de l'information de bureau;
- véhicule cette information.

L'objet DSL le plus adéquat semble donc être l'objet "MESSAGE". On peut trouver sa définition à l'annexe 2.

Comment décrire le document du modèle proposé à l'aide de l'objet DSL "MESSAGE" ?

Schématiquement, la solution se présenterait comme suit:



- On peut définir le document globalement au moyen de la clause DEFINE MESSAGE.
  Exemple: DEFINE MESSAGE lettre-contact.
- Chacun des attributs du document du modèle proposé sera décrit au moyen de la clause CONSISTS OF.
  L'attribut EXPLICATION-TYPE est facultatif; il donne le type du document quand il ne s'agit pas d'un type standard. Cette explication apparait comme un commentaire. C'est pourquoi, on préfèrera la décrire au moyen de la clause DESCRIPTION.

```
Exemple: DEFINE MESSAGE  lettre-acceptation;
           DESCRIPTION;
               explication-type;
           CONSISTS OF  type-document,
                        identifiant-doc,
                        sujet-doc,
                        corps-doc,
                        échéance-doc,
                        état-doc,
                        statut-doc,
                        paragraphes-statiques,
                        support-doc;
```

- Pour la description proprement dite des attributs, on fera appel aux objets DSL "ELEMENT" ou "GROUP".
  - On définira globalement les attributs au moyen des clauses DEFINE ELEMENT et DEFINE GROUP.
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.

  Exemple: pour l'attribut type-document, on aura:

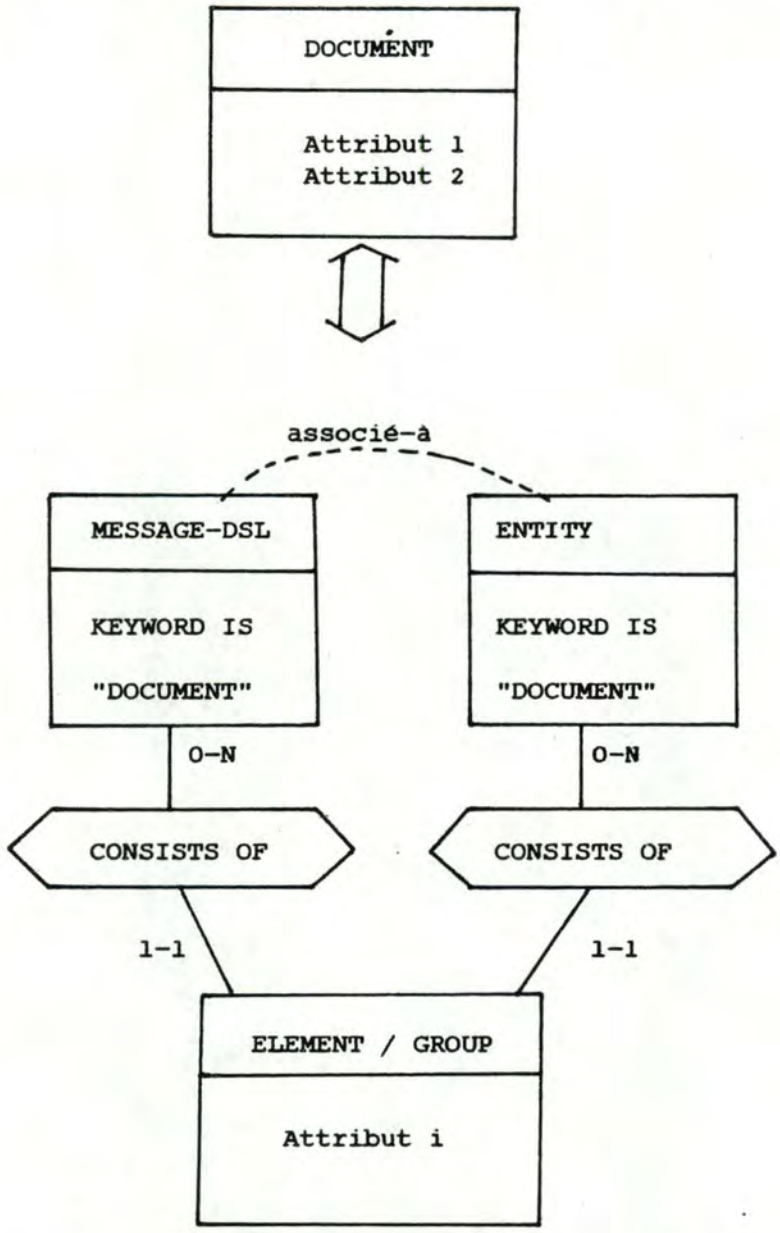
```
           DEFINE ELEMENT type-document;
             DOMAIN OF VALUE ARE  lettre;
```

- Sachant que, dans les résultats, on voudrait pouvoir obtenir une liste de tous les documents décrits par l'utilisateur, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type de l'objet décrit.

```
Exemple: DEFINE MESSAGE lettre-contact;
           KEYWORD ARE  "DOCUMENT";
```

Si nous regardons la description du document dans le modèle proposé, on voit qu'il existe des liens avec d'autres objets informationnels (objets structurants, réponses attendues) et avec l'organisation (expéditeur, destinataires). On présentera la description de ces liens plus tard, mais il ne faut pas oublier qu'ils existent et qu'on devra pouvoir les représenter. Or, par définition de DSL, on ne peut établir aucun lien entre l'objet DSL "MESSAGE" et d'autres objets (il n'existe pas de clause RELATES ou RELATED BY). C'est pourquoi, en plus de décrire les documents au moyen de l'objet DSL "MESSAGE", on recourra également à l'objet DSL "ENTITY".

Schématiquement, cette amélioration de la première solution proposée peut se représenter comme suit:



En plus de la représentation au moyen de l'objet DSL "MESSAGE":

- On définira globalement le document du modèle proposé au moyen de la clause DEFINE ENTITY.
- Les attributs seront les mêmes que pour l'objet DSL "MESSAGE". On les décrira au moyen de la clause CONSISTS OF sans les redécrire de façon individuelle.
- Dans la clause KEYWORD, on décrira encore le type de l'objet.
- Afin de faire le lien entre la représentation du document du modèle proposé à l'aide de l'objet DSL "MESSAGE" et sa représentation à l'aide de l'objet DSL "ENTITY", on utilisera la clause ATTRIBUTE IS pour définir l'attribut "est-associé-à".

En DSL, on décrira donc le document lettre-contact de la façon suivante:

```
/* représentation du document à l'aide de l'objet DSL "ENTITY" */

DEFINE ENTITY lettre-contact-ENT;
  KEYWORD ARE "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              état-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;

/* représentation du document à l'aide de l'objet DSL "MESSAGE" */

DEFINE MESSAGE lettre-contact-MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "lettre-contact-ENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;

/* représentation des attributs */

DEFINE ELEMENT type-lettre-contact;
  DOMAIN OF VALUE ARE lettre;

DEFINE ELEMENT identifiant-lettre-contact;

DEFINE ELEMENT .....
```

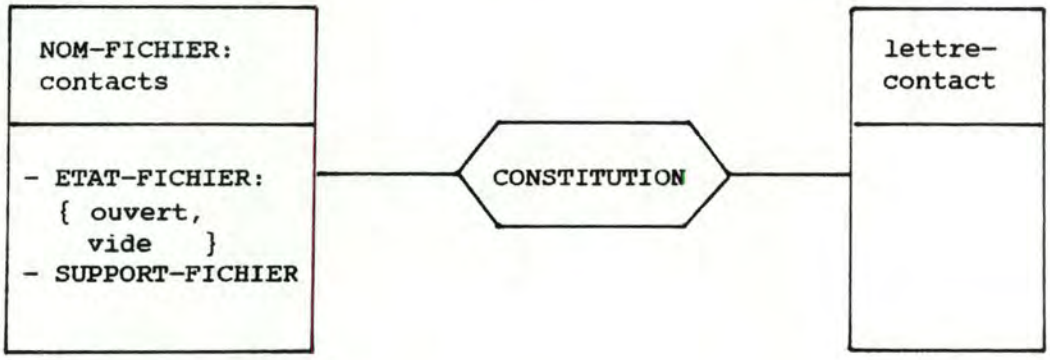
Cette solution possible de transformation que nous venons de décrire pour un objet élémentaire est-elle encore valable en ce qui concerne les objets structurants ? N'existerait-il pas d'autres objets DSL plus adaptés ? Pour répondre à ces questions, analysons la transformation d'un objet structurant: le fichier.

A.1.2. DESCRIPTION D'UN OBJET STRUCTURANT: LE FICHIER

Prenons à titre d'exemple le fichier contacts, qui est un fichier de documents lettre-contact.

* LE FICHIER DANS LE MODELE PROPOSE

Dans le modèle proposé,on avait:



* LE FICHIER EN DSL

Les éléments structurants étant des ensembles d'objets élémentaires, la première idée est de choisir l'objet DSL "SET" pour les représenter.
On aurait donc en DSL:

```
DEFINE SET contacts;
  KEYWORD ARE "FICHIER";
  COLLECTION OF lettre-contact;
```

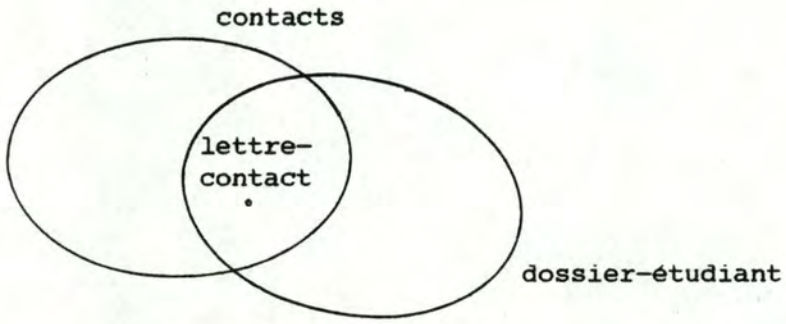
Cependant, si l'objet DSL "SET" permet de grouper facilement des objets, il pose des problèmes dans la représentation des objets structurants de notre modèle:

- L'objet "SET" n'offre aucun moyen de traduire les attributs ETAT et SUPPORT du fichier;
- Tout objet défini comme faisant partie d'un SET y est contenu dès sa création. Ainsi, si on déclare le fichier contacts comme un "SET", toute lettre-contact sera classée dans ce fichier depuis sa création jusqu'à sa destruction. Ces contraintes ne correspondent pas à la philosophie de notre modèle. En effet, prenons l'exemple de la lettre de prise de contact lettre-contact:
  Dans notre modèle, la lettre de prise de contact est représentée par un seul objet informationnel, lettre-contact, dont l'état peut varier au cours du temps. Ainsi, la lettre de prise de contact sera d'abord traitée pour l'envoi d'une réponse à l'étudiant. Elle sera ensuite classée dans le fichier contacts jusqu'à l'ouverture d'un dossier pour l'étudiant. A ce moment, on l'extrait du fichier contacts pour la ranger dans le dossier de l'étudiant.

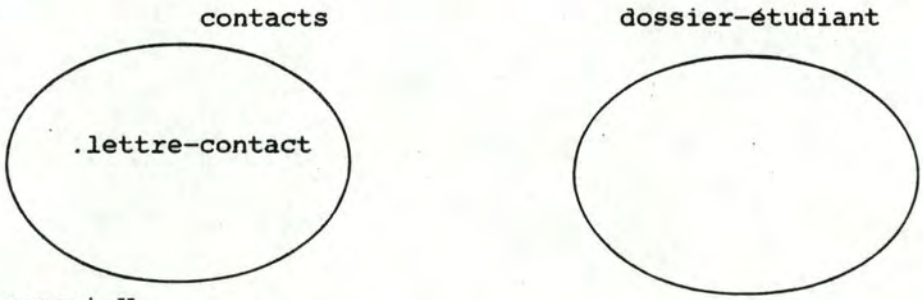
Si on décrit le fichier contacts et le dossier étudiant comme des objets de type "SET", on aura:

```
DEFINE SET contacts;
  COLLECTION OF lettre-contact;
DEFINE SET dos-étudiant;
  COLLECTION OF ....., lettre-contact,
                ....;
```

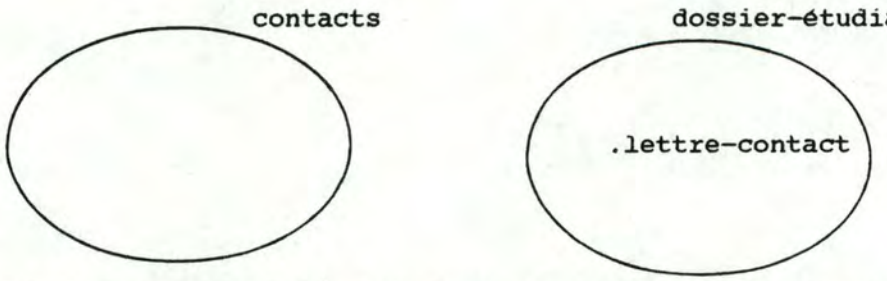
ou schématiquement:



Dès sa création et jusqu'à sa destruction, la lettre de prise de contact appartiendra aux deux ensembles.
Selon le modèle proposé, on devrait avoir:
Au moment X:



Au moment Y:

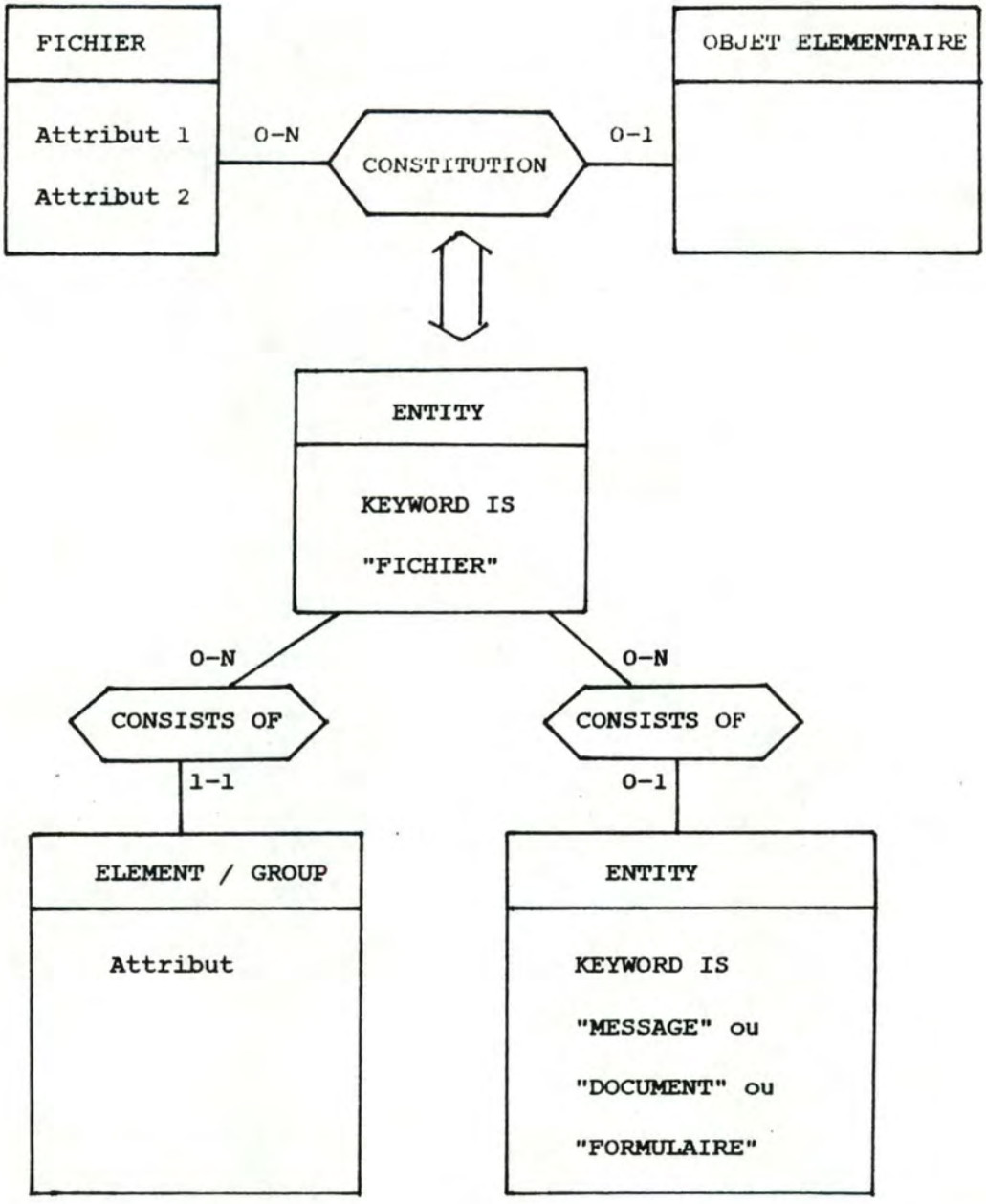


L'objet DSL "SET" ne permet pas d'ajouter ou d'extraire, à n'importe quel moment, un de ses constituants. Décrire un objet informationnel structurant par l'objet DSL "SET" n'est donc pas une bonne solution.

Il faudrait un autre objet DSL permettant:
- de représenter les attributs du fichier;
- de représenter des relations temporaires entre un fichier et ses constituants.

On va donc décrire le fichier au moyen de l'objet DSL "ENTITY".

Cette solution peut se présenter schématiquement comme suit:



- Le fichier peut être décrit globalement au moyen de la clause DEFINE ENTITY.
- Chacun des attributs du fichier sera décrit au moyen de la clause CONSISTS OF.
- Pour la description proprement dite des attributs du fichier, on fera appel aux objets DSL "ELEMENT" et "GROUP".
  - On définira globalement les attributs du fichier au moyen des clauses DEFINE ELEMENT ou DEFINE GROUP.
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.
- Sachant que, dans la liste des résultats, on voudrait pouvoir obtenir une liste de tous les fichiers, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type d'objet décrit.
- Connaissant les objets élémentaires constituant le fichier, on traduira cette contenance par des relations établies entre le fichier et les objets élémentaires qu'il contient.

On décrira donc le fichier contacts de la façon suivante:

```
DEFINE ENTITY contacts;
  KEYWORD ARE "FICHIER";
  CONSISTS OF état-fichier-contact,
              support-fichier-contacts;

DEFINE RELATION lettre-contact-CONSTITUTION-contacts;
 RELATES  lettre-contact
   AS est-contenu-dans
   WITH CONNECTIVITY  0-1;
 RELATES  contacts
   AS contient
   WITH CONNECTIVITY  0-N;
```

Remarque:

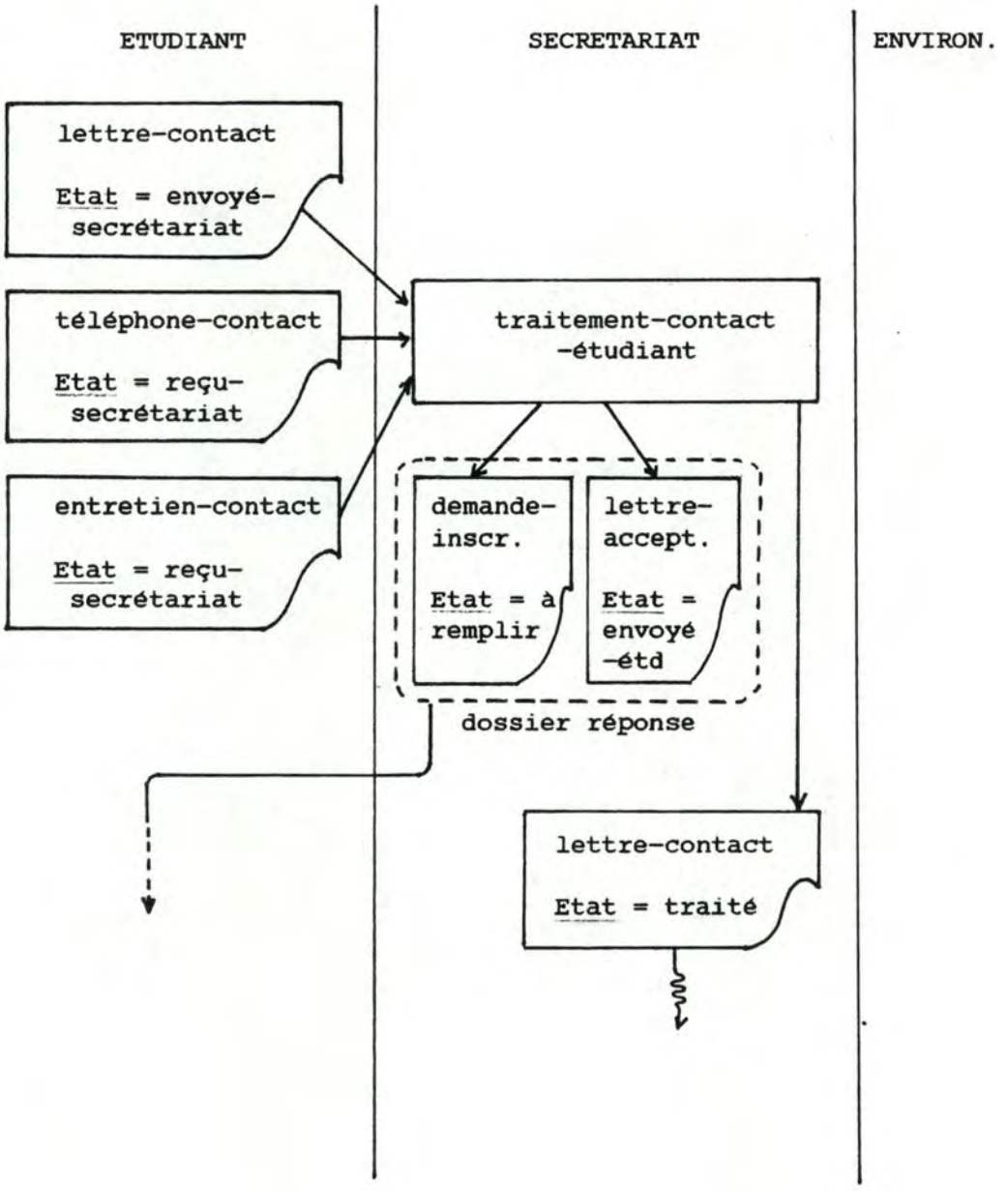
Dans notre modèle, l'objet DSL "ENTITY" va jouer de nombreux rôles. Par rapport à la définition donnée par F. Bodart dans [BOD], l'entité ne représentera plus seulement une chose abstraite ou concrète. Ici, elle sert aussi à représenter le concept d'ensembles de choses.

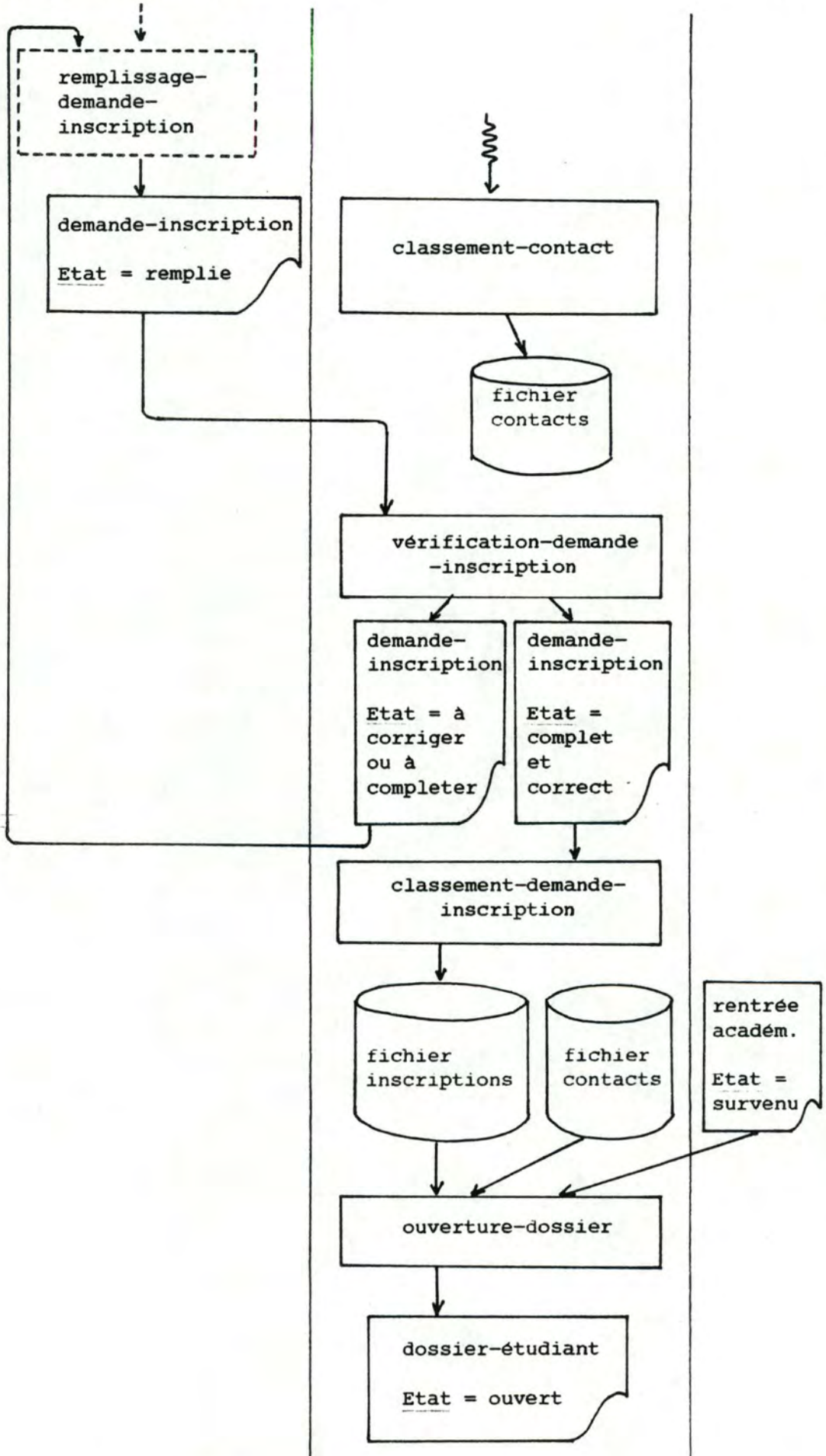
A.2. LA DESCRIPTION STATIQUE DU DIAGRAMME DE FLUX

Tous les objets informationnels que l'on a décrits devront pouvoir être insérés dans un diagramme de flux, de façon à décrire les entrées et les sorties des différentes tâches de bureau ainsi que l'origine et la destination des objets informationnels.
La solution élaborée jusqu'à présent permet-elle d'ajouter ces informations à celles déjà connues ?

Diagramme du flux administratif pris en exemple

Nous choisirons, pour représenter le flux, la représentation proposée par F.Bodart dans [BOD].




remplissage-
demande-
inscription
demande-inscription
Etat = remplie
classement-contact
fichier
contacts
vérification-demande
-inscription
demande-
inscription
Etat = à
corriger
ou à
completer
demande-
inscription
Etat =
complet
et
correct
classement-demande-
inscription
fichier
inscriptions
fichier
contacts
rentrée
académ.
Etat =
survenu
ouverture-dossier
dossier-étudiant
Etat = ouvert

Remarque: Les tâches indiquées dans les rectangles en pointillés sont des tâches extérieures au système étudié, c'est-à-dire au secrétariat.

Nous ne nous attarderons pas trop à la description complète du flux car notre travail se borne à la représentation des objets informationnels et non des tâches.

Nous regarderons seulement si la description des objets informationnels apparaissant dans le diagramme de flux est possible.

Les clauses GENERATED BY et RECEIVED BY de l'objet DSL "MESSAGE" ainsi que les clauses GENERATES ET RECEIVES de l'objet DSL "PROCESS" permettent de décrire l'origine et la destination des objets informationnels.

Les clauses ADDS, DERIVES, MODIFIES travaillant sur les objets ENTITY, ELEMENT, GROUP et RELATION permettent de décrire les entrées et les sorties d'un traitemnt.

A.4. RESUME

La description de chacun des objets informationnels du modèle proposé, pris individuellement, au niveau statique, donne le tableau suivant:

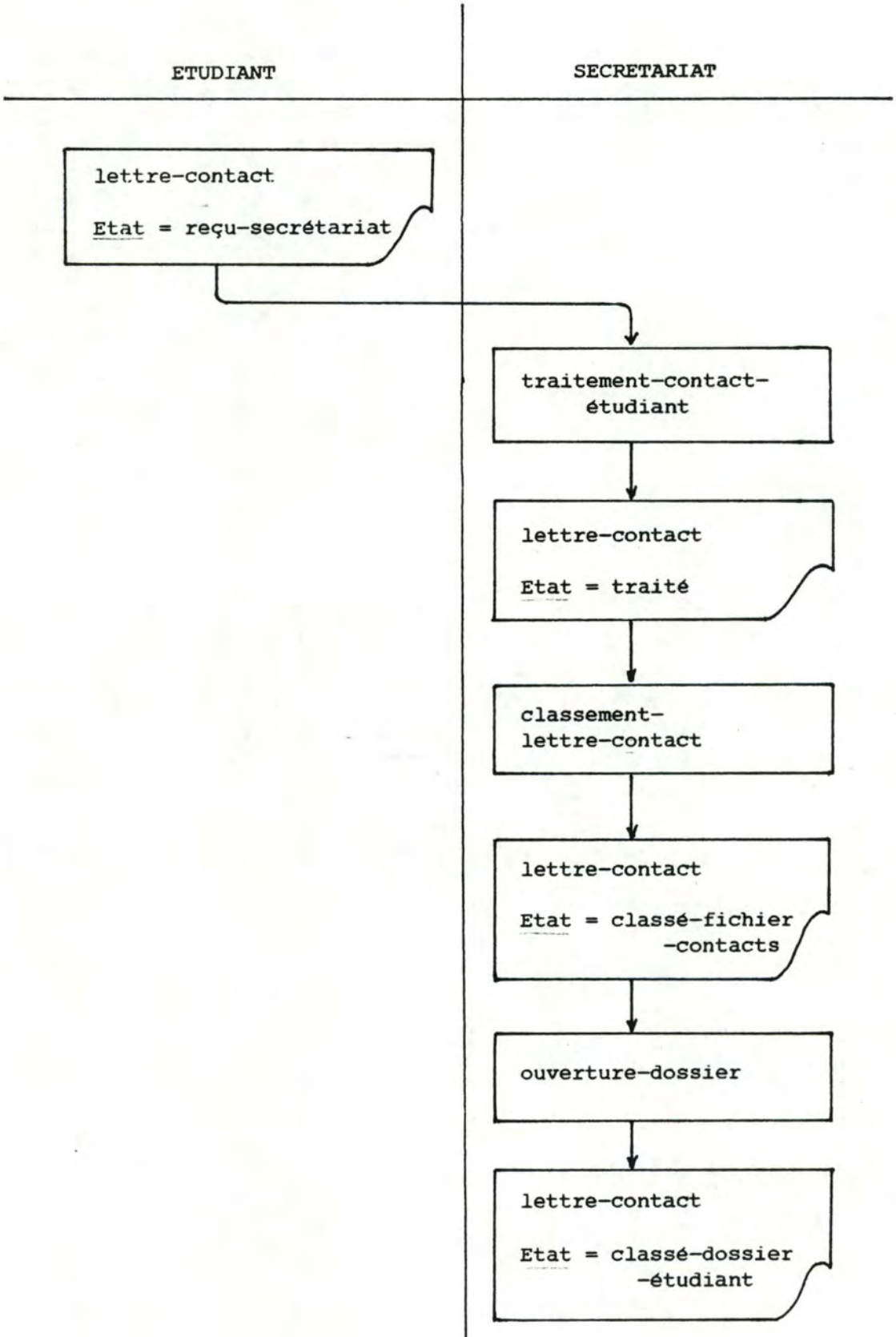
OBJETS INFORMATIONNELS DU MODELE PROPOSE	TRANSFORMATION EN DSL
OBJETS ELEMENTAIRES (message, document, formulaire)	ENTITY + MESSAGE
OBJETS STRUCTURANTS (dossier, fichier, pile)	ENTITY + RELATION avec les constituants

B. LE NIVEAU DYNAMIQUE

Jusqu'à présent, nous avons vu comment décrire les objets informationnels à un niveau statique, c'est-à-dire purement descriptif.

Lorsqu'on fera intervenir la dynamique, c'est-à-dire lorsqu'on voudra simuler le comportement du système décrit à l'aide du modèle, la solution choisie suffira-t-elle toujours ?

Examinons la dynamique du petit flux servant d'exemple de base, et plus particulièrement la dynamique liée au document lettre-contact.
Lorsque l'étudiant envoie sa lettre de prise de contact, la secrétaire la traite, c'est-à-dire qu'elle prépare une réponse pour l'étudiant. La lettre-contact traitée est ensuite classée dans un fichier: le fichier contacts. A la rentrée académique, la secrétaire reprend cette lettre de contact pour la ranger dans le dossier de l'étudiant.


ETUDIANT
SECRETARIAT
lettre-contact
Etat = reçu-secrétariat
traitement-contact-étudiant
lettre-contact
Etat = traité
classement-lettre-contact
lettre-contact
Etat = classé-fichier-contacts
ouverture-dossier
lettre-contact
Etat = classé-dossier-étudiant

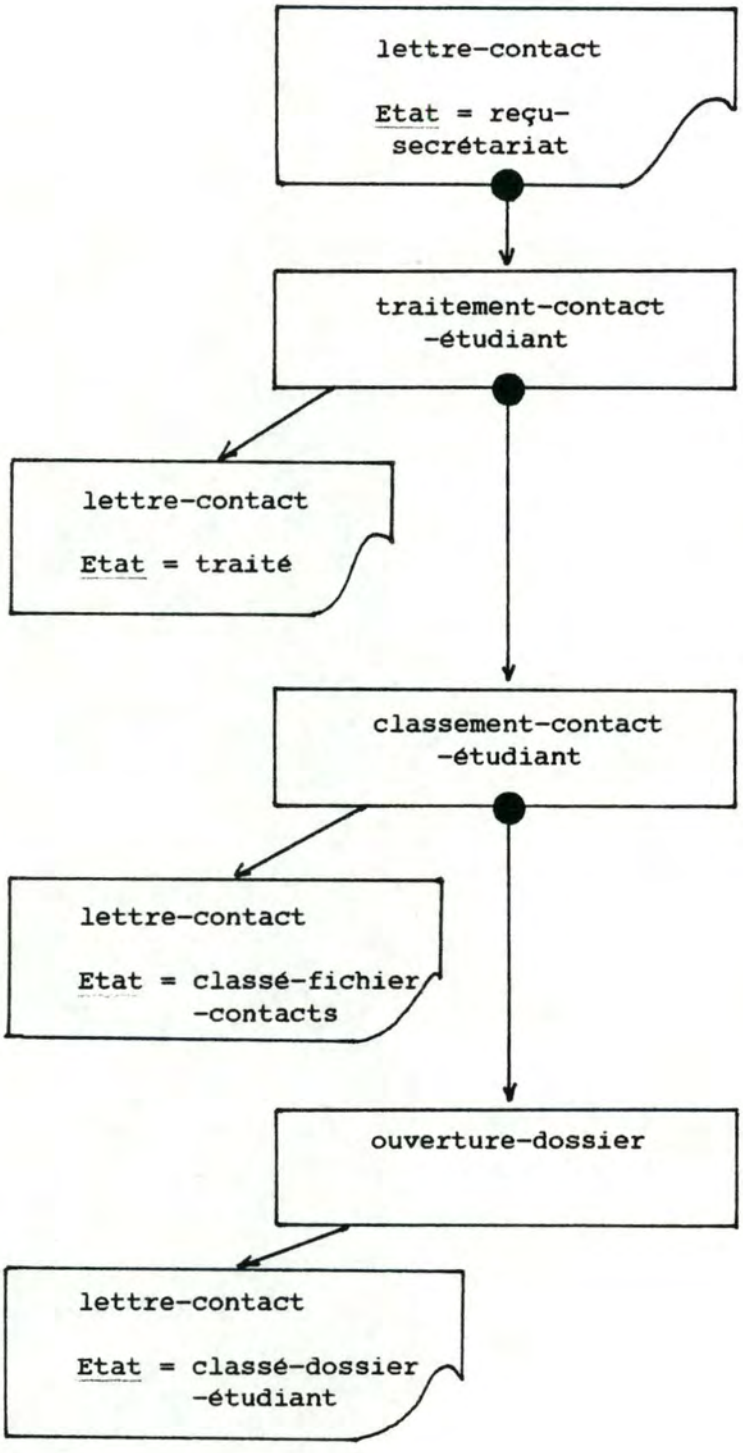
En DSL, la dynamique se traduira par:

```
DEFINE MESSAGE lettre-contact;
  CONSISTS OF type,
              identifiant,
              sujet,
              corps,
              échéance,
              état,
              statut,
              paragraphes-statiques,
              support.

ON GENERATION TRIGGERS traitement-contact-étudiant
                  IF état = reçu-secrétariat;

ON GENERATION TRIGGERS classement-contact
                  IF état = traité;

ON GENERATION TRIGGERS ouverture-dossier
                  IF état = classé-fichier-contacts;
```

Si on garde toute cette information d'un point de vue descriptif, il n'y a aucun problème: on peut décrire les conditions de déclenchement qui portent sur l'état du message DSL représentant le document lettre-acceptation.
Mais si on veut simuler le système modélisé, ces conditions de déclenchement ne sont plus valables. En effet, dans le programme de simulation du système IDA, les conditions de déclenchement sont traduites par des conditions probabilistes; elles ne peuvent porter sur un constituant du message DSL.
En simulation, la dynamique décrite ci-dessus se réduira donc à:

```
DEFINE MESSAGE  lettre-contact;
  ON GENERATION TRIGGERS traitement-contact-étudiant
                  WITH PROBABILITY x;
  ON GENERATION TRIGGERS classement-contact
                  WITH PROBABILITY y;
  ON GENERATION TRIGGERS ouverture-dossier
                  WITH PROBABILITY z;
```

La génération d'un document lettre-contact (qu'il soit reçu par le secrétariat, traité ou classé dans le fichier contacts) entrainera donc l'exécution simultanée des tâches traitement-contact-étudiant, classement-contact et ouverture-dossier, puisqu'il n'y a plus de distinction entre une lettre-contact reçue par le secrétariat, une lettre-contact traitée et une lettre-contact classée.
Si on veut garder la solution initiale, il faudra donc renoncer à la dynamique par les messages pour passer à une dynamique par les processus:



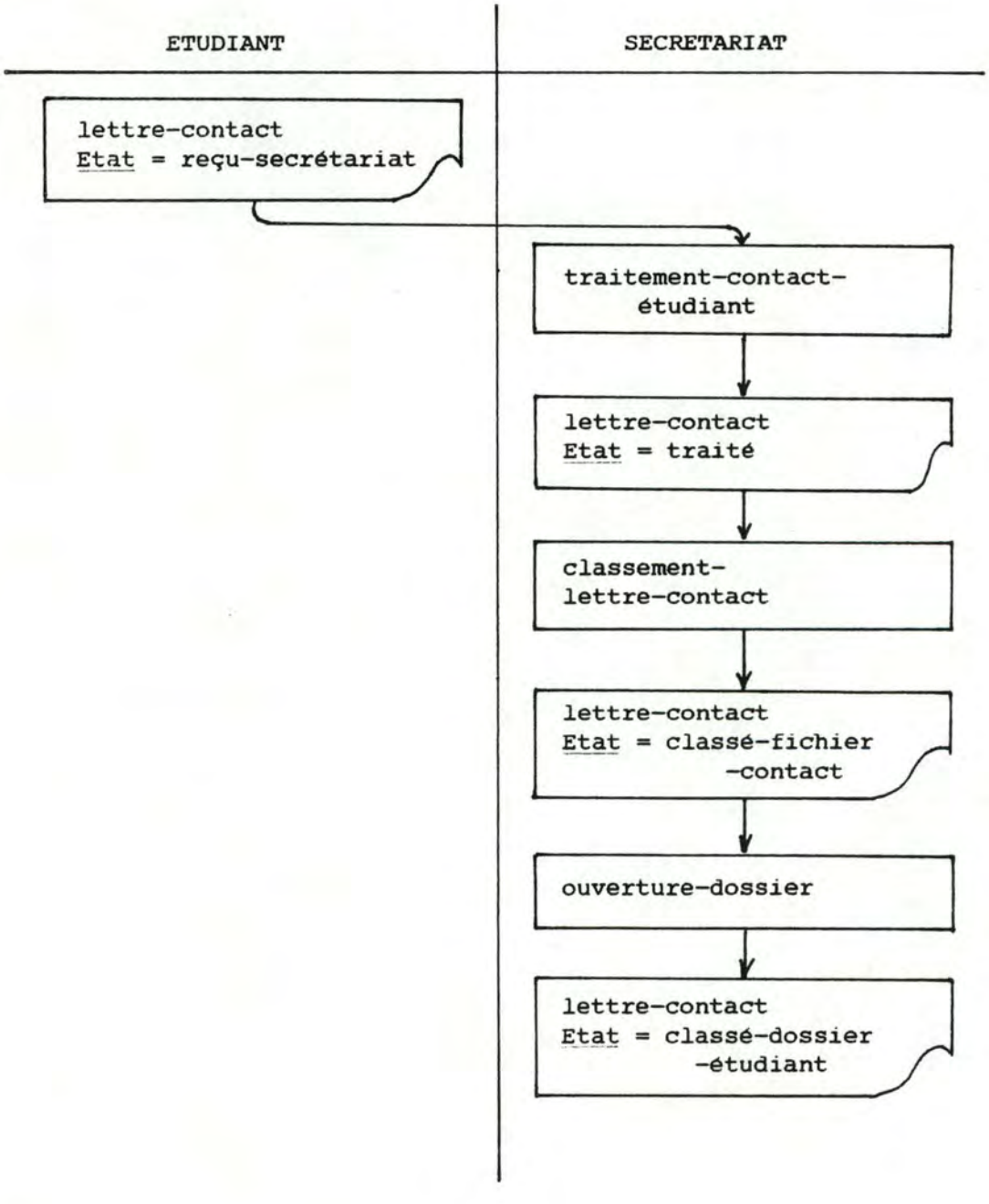
Le classement de la lettre-contact se fera après la terminaison du traitement-contact-étudiant et l'ouverture du dossier se fera après la terminaison de classement-contact.

Ainsi, du point de vue de la dynamique, la solution choisie ne permet pas d'adopter la dynamique par les messages. On doit décrire le flux au moyen de la dynamique par les processus, comme cela se fait dans IDA. L'inconvénient de cette dynamique est qu'on perd le concept d'états successifs des objets, proposé dans le MIB.

Comment pouvons-nous corriger la solution possible de représentation du modèle proposé en DSL à laquelle nous étions arrivés pour pouvoir utiliser la dynamique par les messages DSL et ainsi pouvoir simuler le comportement du modèle décrit ?

Réponse: "A un objet du modèle proposé correspond un objet DSL "ENTITY" et autant d'objets DSL "MESSAGE" qu'il y a d'états successifs dans le cycle de vie de l'objet du modèle proposé."

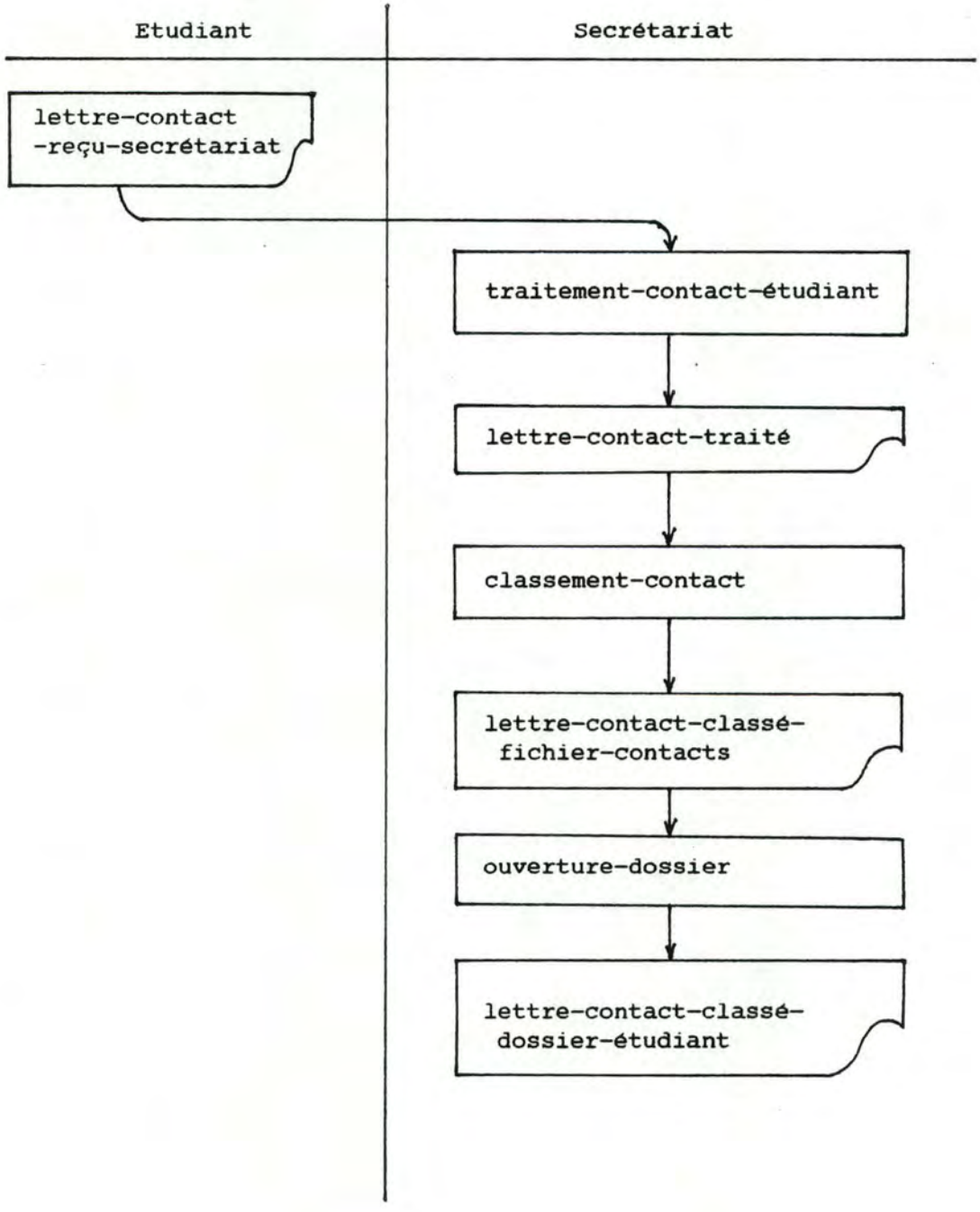
Reprenons le schéma de flux lié au document "lettre-contact":



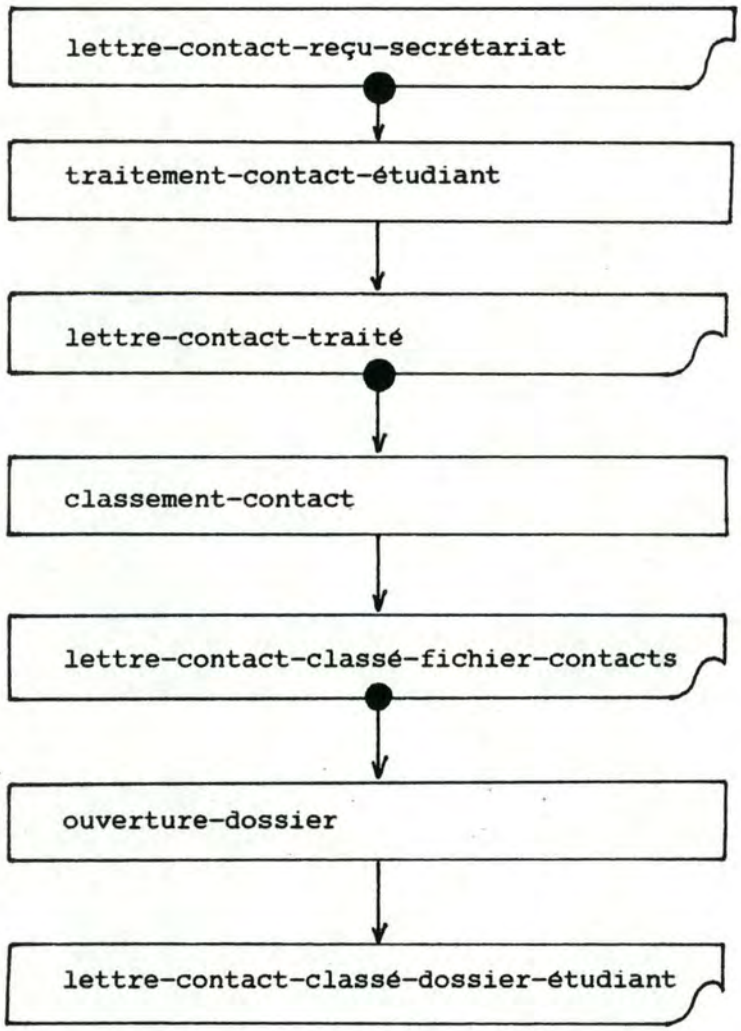
Si on veut pouvoir employer la dynamique par les messages, il faut trouver le moyen de différencier la lettre de prise de contact dans l'état reçu-secrétariat, dans l'état traité et dans l'état classé-fichier-contact. La seule façon de les différencier est d'en faire trois objets DSL différents.
Ainsi, la nouvelle solution reviendra à:

- représenter l'objet du modèle proposé par un objet DSL "ENTITY". On gardera l'attribut "état de l'objet" à cet endroit, ce qui nous permettra de conserver l'idée du cycle de vie de l'objet adopté dans le modèle proposé.
- représenter l'objet du modèle proposé par autant d'objets DSL "MESSAGE" qu'il y a d'états dans le cycle de vie de l'objet du modèle proposé.

Ainsi, si on reprend l'exemple du flux lié à la lettre de prise de contact, on aura:



ce qui donne le schéma de déclenchement:



et le texte DSL:

```
DEFINE MESSAGE lettre-contact-reçu-secrétariat;
  ON GENERATION TRIGGERS traitement-contact-étudiant;
DEFINE MESSAGE lettre-contact-traité;
  ON GENERATION TRIGGERS classement-contact;
DEFINE MESSAGE lettre-contact-classé-fichier-contacts;
  ON GENERATION TRIGGERS ouverture-dossier;
```

Il n'y a plus de condition de déclenchement portant sur un constituant de l'objet DSL "MESSAGE" et qui ne pouvait être vérifiée à la simulation. Chaque message déclencheur est distinct. La dynamique par les messages peut être récupérée.

La représentation du document lettre-contact à l'aide de cette nouvelle solution devient:

- On représente la description du document lettre-contact au moyen de l'objet DSL "ENTITY".

```
DEFINE ENTITY lettre-contact;
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              état-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;
```

- On représente également la description du document lettre-contact par autant d'objets DSL "MESSAGE" qu'il y a d'états dans son cycle de vie.

```
DEFINE MESSAGE lettre-contact-envoyé-secrétariat;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;


DEFINE MESSAGE lettre-contact-traité;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;
```

```
DEFINE MESSAGE lettre-contact-classé-fichier-contacts;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;


DEFINE MESSAGE lettre-contact-classé-dossier-étudiant;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;
```

Remarque:

La transformation ci-dessus consiste à redécrire les objets DSL "MESSAGE" liés à l'objet de façon systématique, en les traitant tous de la même manière.

Le langage DSL permet cependant d'éviter cette redondance en utilisant la clause SAME DATA STRUCTURE AS qui traduirait en quelque sorte l'héritage de propriétés entre les différents objets DSL "MESSAGE" liés à un même objet et permettrait de ne pas reprendre systématiquement la description de tous les attributs.

Si nous utilisons cette clause, la représentation du document lettre-contact se réduirait à:

```
DEFINE ENTITY lettre-contact;
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              état-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;
```

```
DEFINE MESSAGE lettre-contact-envoyé-secrétariat;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  CONSISTS OF type-lettre-contact,
              identifiant-lettre-contact,
              sujet-lettre-contact,
              corps-lettre-contact,
              échéance-lettre-contact,
              statut-lettre-contact,
              paragraphes-statiques-lettre-contact,
              support-lettre-contact;

DEFINE MESSAGE lettre-contact-traité;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  SAME STRUCTURE AS lettre-contact-envoyé-secrétariat;

DEFINE MESSAGE lettre-contact-classé-fichier-contacts;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  SAME STRUCTURE AS lettre-contact-envoyé-secrétariat;

DEFINE MESSAGE lettre-contact-classé-dossier-étudiant;
  ATTRIBUTE IS "associé-à"  "lettre-contact";
  KEYWORD IS "DOCUMENT";
  SAME STRUCTURE AS lettre-contact-envoyé-secrétariat;
```

C. CONCLUSION

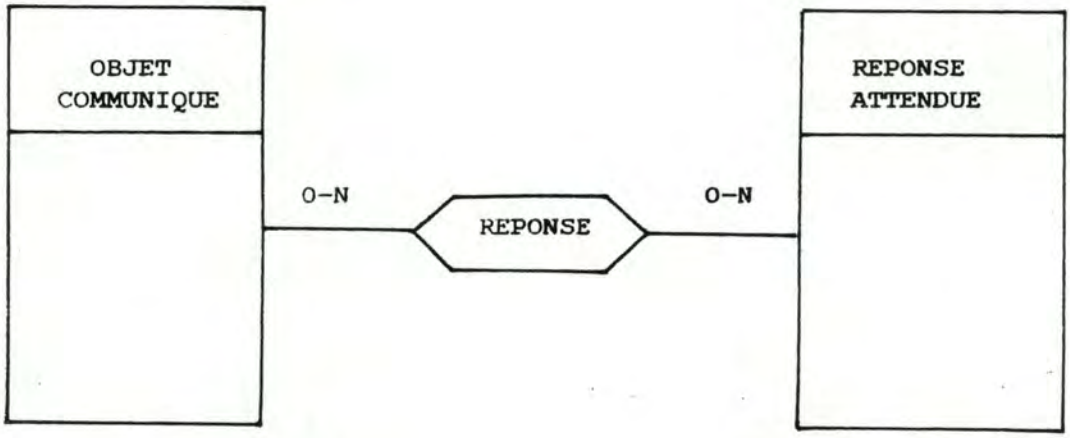
Le raisonnement suivi dans la recherche d'une solution possible de la transformation des objets informationnels proprement dits nous a conduit à la solution suivante:
" A un objet du modèle proposé correspond un objet DSL "ENTITY" et autant d'objets DSL "MESSAGE" qu'il y a d'états dans son cycle de vie. "

Du point de vue descriptif, cette solution a l'inconvénient de multiplier les objets DSL à représenter pour décrire les objets du modèle proposé. En effet, un objet DSL "ENTITY" et un objet DSL "MESSAGE" auraient suffi.

Cependant, du point de vue de la dynamique, cette solution nous permet de nous servir de la dynamique par les messages DSL et donc de conserver le concept d'état de l'objet décrit dans le modèle proposé.

3.4.2.2 LA REPRESENTATION DE LA DESCRIPTION DES REPONSES ATTENDUES

Chacune des réponses attendues suite à la communication d'un objet sera décrite dans le système d'information par un objet DSL "ENTITY" et un ou plusieurs objets DSL "MESSAGE", puisque ces réponses sont des objets informationnels. Il suffira d'établir un lien entre l'objet communiqué et les objets attendus en réponse. En DSL, on peut représenter ce lien au moyen de l'objet DSL "RELATION", liant l'objet DSL "ENTITY" représentant l'objet communiqué et l'objet DSL "ENTITY" représentant une réponse attendue.



Exemple:
Le document lettre-contact attend en réponse les formulaires lettre-acceptation et demande-inscription.
Tous ces objets auront été décrits.
Il restera à décrire la relation entre lettre-contact et lettre-acceptation et la relation entre lettre-contact et demande-inscription.
On aura donc en DSL:

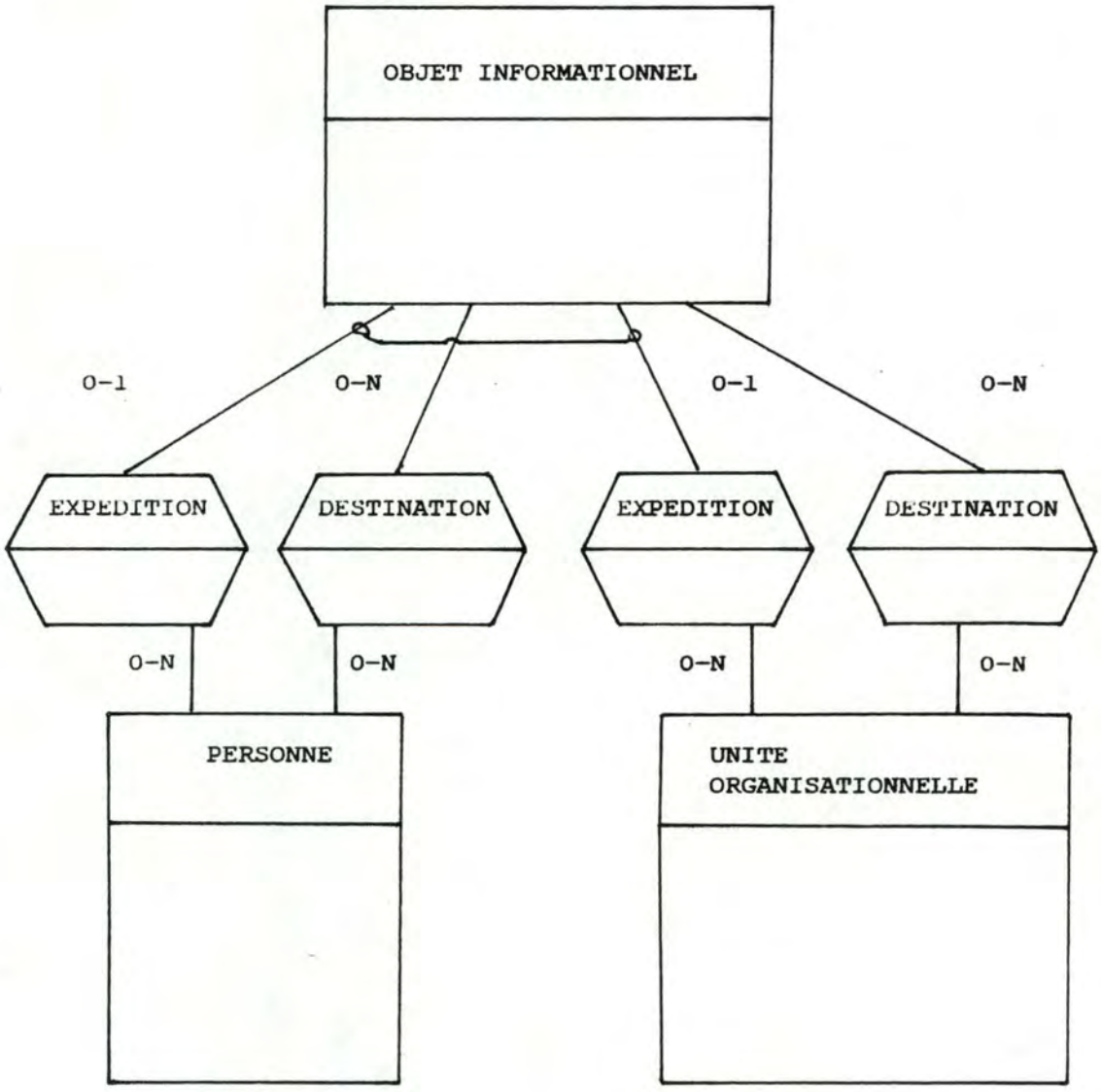
```
DEFINE RELATION lettre-acceptation-REPONSE-lettre-contact;
  RELATES lettre-acceptation
    AS répond-à
    WITH CONNECTIVITY  0-N;
  RELATES lettre-contact
    AS a-pour-réponse
    WITH CONNECTIVITY  0-N;

DEFINE RELATION demande-inscription-REPONSE-lettre-contact;
  RELATES demande-inscription
    AS répond-à
    WITH CONNECTIVITY  0-N;
  RELATES lettre-contact
    AS a-pour-réponse
    WITH CONNECTIVITY  0-N;
```

3.4.2.3 LA REPRESENTATION DE DESCRIPTION DES LIENS AVEC L'ORGANISATION

Dans le modèle proposé, nous avons établi des liens entre les objets informationnels et l'organisation. Ces liens spécifient:

- quel est l'expéditeur de l'objet informationnel;
- quels sont les différents destinataires de cet objet.

Dans notre modèle, nous avions représenté ces liens par des associations entre l'objet informationnel et l'organisation:



Comment représenter ces concepts en DSL ?

L'objet qui nous permet de décrire l'organisation en DSL est l'objet "INTERFACE". L'interface est l'origine ou la destination des objets "MESSAGE" en DSL.

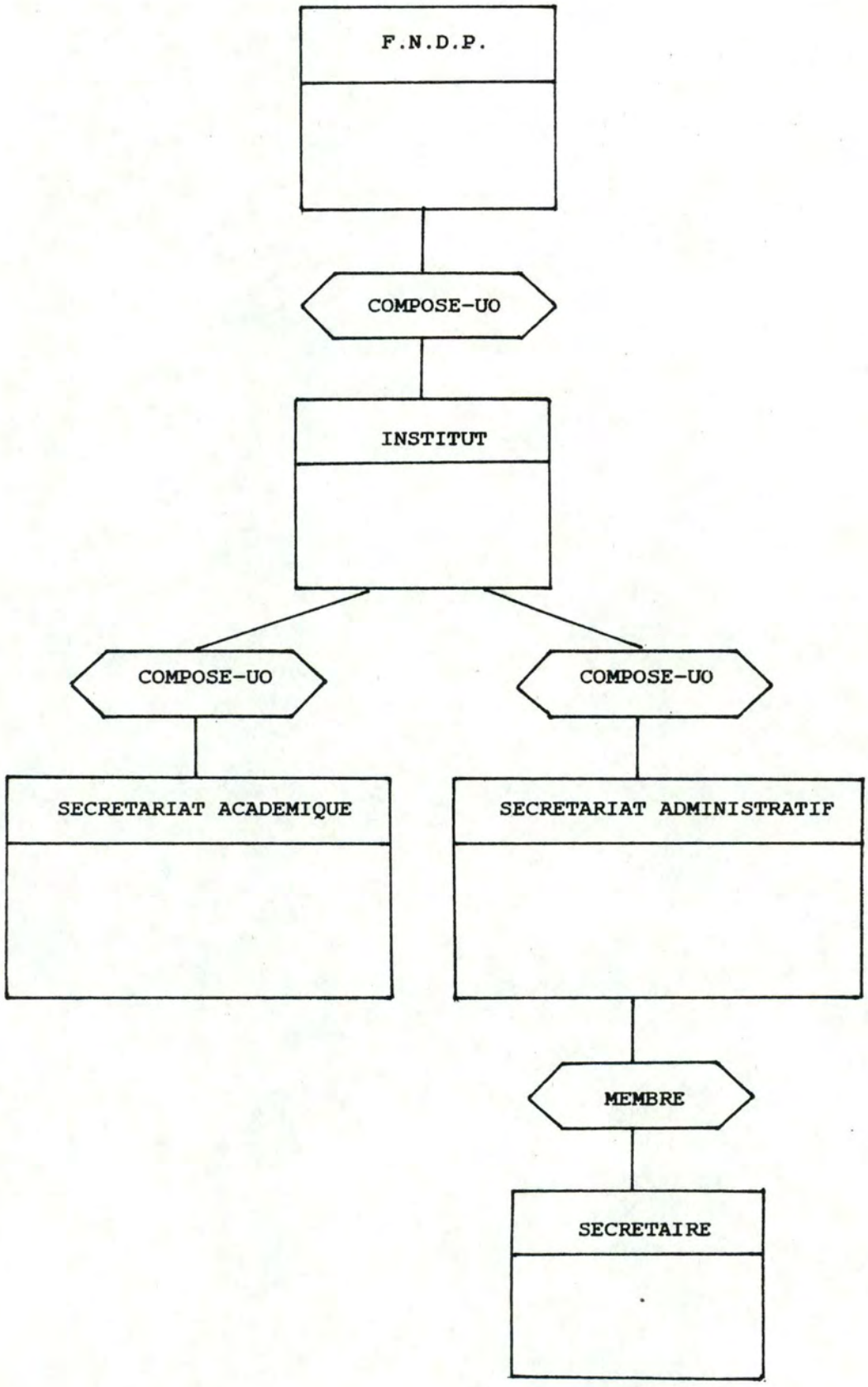
Que nous offre l'objet DSL "INTERFACE" ?
- L'objet "INTERFACE" nous permet de définir un service de l'organisation.
- On peut décrire les objets informationnels qu'il génère ou qu'il reçoit.
- On peut décrire les tâches qu'il gère.

Cependant, cela ne suffit pas pour représenter tous les concepts apparaissant dans le modèle proposé.
En effet:
- Dans le modèle proposé, on a décrit finement la structure de l'organisation: l'organisation est décomposée en unités organisationnelles; les unités organisationnelles se décomposent elles-mêmes en unités organisationnelles ou en personnes qui peuvent en être responsables.
  L'objet "INTERFACE" ne nous permet pas une description aussi fine de l'organisation.
- Dans le modèle proposé, on retient la date d'expédition d'un objet informationnel.
  L'objet "INTERFACE" permet de décrire la génération d'un objet informationnel, mais pas sa date de génération, c'est-à-dire sa date d'expédition à un tiers.

Comment représenter l'organisation aussi finement que dans le modèle proposé ?

Nous ne nous attacherons pas, dans cette partie, à décrire l'organisation en détail. Cette description détaillée de l'organisation pourrait être réalisée en DSL au moyen des objets "ENTITY" et "RELATION":
- La description de l'organisation, des unités organisationnelles et des personnes peut être représentées par l'objet "ENTITY".
- La structure existant entre ces différentes parties peut être représentée en DSL par l'objet "RELATION".

Exemple simple:
La société "F.N.D.P." se compose de différentes facultés dont fait partie l'institut d'informatique. L'institut est lui même divisé en différents services, parmi lesquels le secrétariat académique et le secrétariat administratif. La secrétaire travaille dans ce dernier service.


F.N.D.P.
COMPOSE-UO
INSTITUT
COMPOSE-UO
COMPOSE-UO
SECRETARIAT ACADEMIQUE
SECRETARIAT ADMINISTRATIF
MEMBRE
SECRETAIRE

Ayant cette représentation de la description de l'organisation à l'aide des objets DSL "ENTITY" et "RELATION", comment représenter les liens de communication entre les objets informationnels et l'organisation ?
Réponse: Par l'objet DSL "RELATION".
On connait le nom de l'expéditeur et des destinataires.
On connait le nom de l'objet communiqué.
On peut donc définir une "RELATION" entre les deux objets DSL "ENTITY" représentant la description de l'objet communiqué, et celle de l'expéditeur ou de chaque destinataire.

Exemple:
L'étudiant envoie un document lettre-contact au secrétariat pour demander les modalités d'inscription.
On aura déjà décrit les objets informationnels et l'organisation; on aura donc:

```
/* représentation de la description du document lettre-contact */

   DEFINE ENTITY lettre-contact;
     ........

/* représentation de la description de l'organisation */

   DEFINE ENTITY étudiant;
     ........

   DEFINE ENTITY secrétariat;
     ........
```

Pour décrire la communication du document, on écrira:

```
   DEFINE RELATION étudiant-EXPED-lettre-contact;
     CONSISTS OF date-exped-lettre-contact;
     RELATES étudiant
       AS expédie
       WITH CONNECTIVITY 0-N;
     RELATES lettre-contact
       AS est-expédié-par
       WITH CONNECTIVITY 0-1;

   DEFINE RELATION secrétariat-DEST-lettre-contact;
     RELATES secrétariat
       AS reçoit
       WITH CONNECTIVITY 0-N;
     RELATES lettre-contact
       AS est-destiné-à
       WITH CONNECTIVITY 0-N;
```

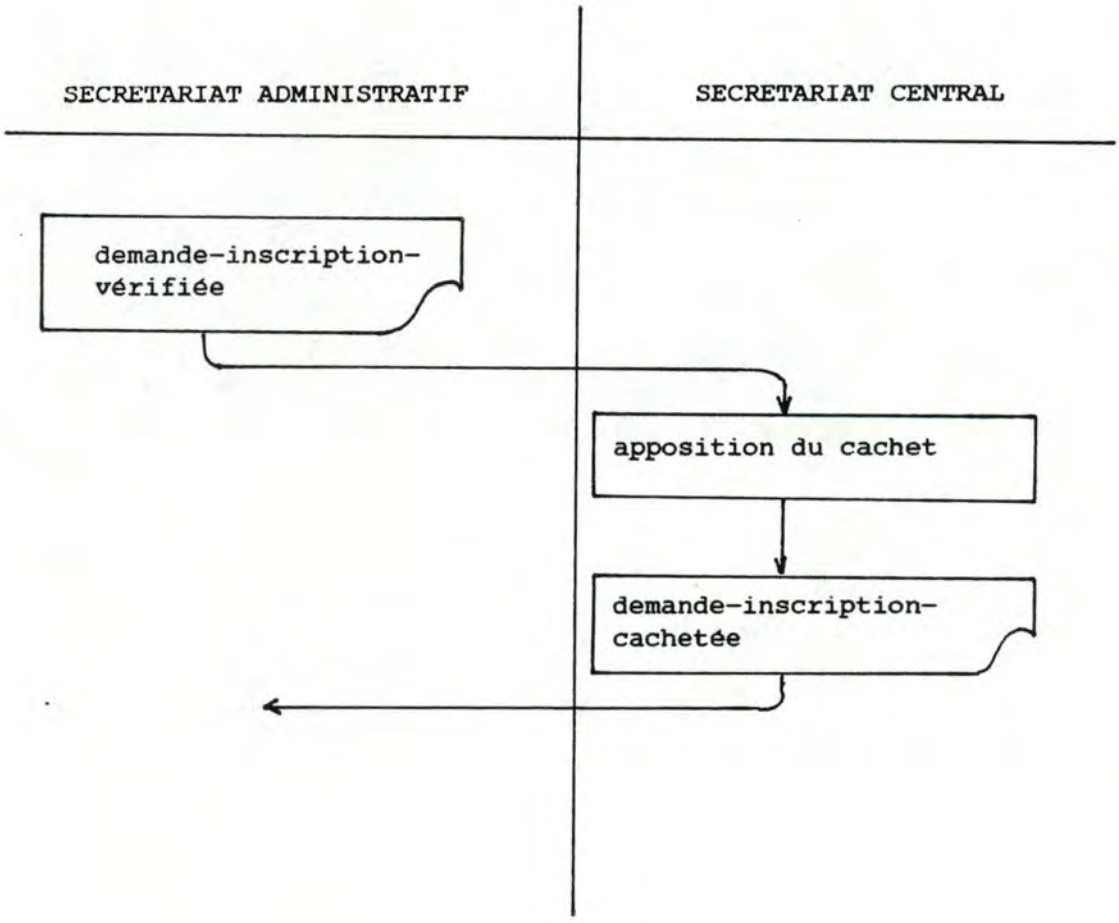
Du point de vue descriptif, cette solution permet donc de mieux décrire l'information contenue dans le modèle proposé que l'objet DSL "INTERFACE".

- On peut décrire finement l'organisation.
- On peut décrire la date d'expédition.
- On sait par qui et à qui les objets informationnels sont communiqués.

Du point de vue dynamique, représenter l'organisation au moyen de l'objet DSL "INTERFACE" n'apporterait rien de plus car les INTERFACES n'interviennent pas dans la dynamique.

Pour simuler le comportement d'un système, il ne faut pas décrire la génération des objets MESSAGE par un INTERFACE, mais bien par les PROCESSUS gérés par cet interface.

Exemple:
Le secrétariat administratif reçoit une demande d'inscription remplie. Il la vérifie puis l'envoie au secrétariat central qui y appose un cachet puis la lui retourne.



Dire que le secrétariat administratif est un interface qui génère une demande d'inscription vérifiée et génère une demande d'inscription cachetée n'apporte rien du point de vue dynamique: en effet, sans décrire de processus, on ne peut dire qu'à chaque demande d'inscription vérifiée correspond une demande d'inscription cachetée.

Pour la dynamique, on décrira donc un processus pouvant être une "boite noire" (on peut ne pas connaitre ce qu'il fait exactement si c'est un processus externe à l'organisation étudiée) qui reçoit une demande d'inscription vérifiée et génère la demande d'inscription cachetée correspondante.

En conclusion:

On représentera les concepts d'expéditeur et de destinataire d'un objet informationnel au moyen des objets DSL "ENTITY" et "RELATION".

En effet, cette solution permet une meilleure représentation du modèle proposé que l'objet "INTERFACE".

3.4.2.4 SYNTHESE: CHOIX D'UNE SOLUTION

Après avoir exposé une solution possible de transformation que nous avons progressivement améliorée, nous pouvons faire une synthèse de la solution qui nous parait la meilleure étant donné les objectifs de documentation et de simulation.

OBJETS DU MODELE PROPOSE	REPRESENTATION EN DSL
L'ORGANISATION:_	
- les objets organisationnels	"ENTITY"
- la structure de l'organisation	"RELATION"
LES OBJETS INFORMATIONNELS:	
- les objets élémentaires	1 "ENTITY" + 1 "MESSAGE" par état du cycle de vie de l'objet élémentaire.
- les objets structurants	1 "ENTITY" + des "RELATIONs" avec les constituants.
LES REPONSES ATTENDUES:	1 "RELATION" entre chaque "ENTITY" représentant les réponses attendues et l' "ENTITY" représentant l'objet informationnel auquel on doit répondre.
L'INTERFACE:	1 "RELATION" entre chacune des "ENTITYs" représentant l'expéditeur et les destinataires et l' "ENTITY" représentant l'objet communiqué.

3.5 IMPLANTATION EN DSL DU MODELE PROPOSE

Après avoir choisi une solution possible de description à l'aide de DSL qui conserve la sémantique du modèle proposé, nous allons maintenant voir comment procéder à la transformation des objets informationnels du modèle proposé en DSL.

3.5.1 COMMENT PROCEDER ? GENERALITES

Nous avons déjà donné des exemples lors du choix d'une solution, mais nous n'avons pas encore vu en détail comment les construire.

- Où va-t-on chercher l'information ?
- Comment construire les spécifications en DSL ?
- Comment sont formés les noms des objets en DSL ?

Nous tenterons de répondre à ces questions.

Nous allons d'abord décrire les principes généraux du processus imaginé. Nous en donnerons un exemple détaillé dans le point suivant, où nous verrons comment représenter les objets informationnels en DSL.

Où va-t-on chercher l'information connue pour le modèle proposé et qui nous permettra d'écrire les spécifications en DSL ?

Comme nous l'avons décrit dans le schéma général de transformation (point 3.1.), l'utilisateur décrira les objets de bureau selon le modèle de structuration des informations, à partir d'écrans de saisie. Après vérification de la cohérence et de la complétude de ces informations, elles seront rangées dans des enregistrements prédéfinis et seront ainsi disponibles pour la transformation en DSL.

Comment construire les spécifications en DSL ?

Pour chaque objet du modèle proposé, nous allons définir des squelettes de texte DSL. Ces squelettes de texte DSL seront écrits sur base de la solution élaborée ci-avant. Il s'agira d'une construction incomplète de texte en DSL.
Par exemple, pour l'objet "ENTITY" représentant un document du modèle proposé, on aura:

```
DEFINE ENTITY ____________;
  KEYWORD ARE "DOCUMENT";
  CONSISTS OF type_____________,
              identifiant_____________,
              sujet_________,
              corps_________,
              état_________,
              échéance_________,
              statut_________,
              paragraphes-statiques_________,
              support_________;
```

Ces squelettes de texte DSL devront être complétés lors de la transformation du modèle proposé en DSL, à l'aide de l'information introduite par l'utilisateur et rangée dans des enregistrements.

Comment former les noms des objets DSL à l'aide de l'information contenue dans les enregistrements ?

En DSL, les noms d'objet doivent être uniques. Nous serons amenés à représenter la description d'un seul objet du modèle proposé par plusieurs objets DSL. Pour distinguer les différentes représentations en DSL d'un même objet du modèle proposé, nous en postfixerons le nom.

En DSL, les noms doivent se composer de 30 lettres au maximum. On devra donc:

- soit limiter la longueur des noms donnés par l'utilisateur;
- soit ne pas limiter leur longueur au départ mais les raccourcir lors de la transformation en DSL.

La solution du raccouci n'est pas possible car si on peut encore passer de la description du modèle proposé en DSL, ayant perdu le nom de l'objet donné initialement par l'utilisateur, on ne pourra plus faire la transformation inverse.

Quelle longueur limite doit-on avoir ?
Si on repense à la solution choisie:

- Certains noms sont postfixés pour faire la distinction entre les objets DSL "MESSAGE" et "ENTITY" représentant le même objet du modèle proposé.
- Dans les relations liant les réponses attendues à l'objet informationnel, ou liant un objet communiqué à l'organisation, on fait apparaître les noms des deux objets reliés et le nom de la relation.

On doit pouvoir représenter, avec 30 caractères, les trois schémas de noms DSL suivants:

OBJET	_ENT

OBJET	ETAT	_MES

OBJET 1	-	RELATION	-	OBJET 2

Il est donc raisonnable de fixer la longueur limite des noms introduits par l'utilisateur à 12 caractères.

3.5.2 LA REPRESENTATION EN DSL DES OBJETS INFORMATIONNELS

Nous allons présenter ici, de façon détaillée, le processus imaginé pour passer de la représentation des objets informationnels du modèle proposé à leur représentation en DSL.

Nous nous bornerons à décrire ici les objets "document" et fichier". La description des autres objets se trouve à l'annexe 4.

3.5.2.1 REPRESENTATION EN DSL DES CONCEPTS COMMUNS AUX OBJETS

Certains attributs des objets n'ont pas de valeur possible ou de valeur connue lors de la spécification du bureau. C'est le cas, par exemple, du support de l'objet. Le support d'un objet est donc défini de la même façon pour tous les objets.
Exemple:
Supposons qu'on ait le document lettre-contact, le formulaire demande-inscription et le fichier contacts.
On aura les descriptions suivantes en DSL:

```
DEFINE ENTITY lettre-contact;
  CONSISTS OF ....., support-lettre-contact, ..... ;

DEFINE ENTITY demande-inscription;
  CONSISTS OF ....., support-demande-inscription, ..... ;

DEFINE ENTITY contacts;
  CONSISTS OF ....., support-contacts, ..... ;
```

et on définira les éléments:

```
DEFINE ELEMENT support-lettre-contact;
  .....

DEFINE ELEMENT support-demande-inscription;
  .....

DEFINE ELEMENT support-contacts;
  .....
```

Pour éviter une multiplication des objets DSL, ces éléments peuvent faire l'objet d'une seule définition, la même pour tous les objets. Plus tard, quand on voudra donner ou rechercher la valeur d'un de ces éléments, il suffira de le qualifier: < nom-élément > . < nom-objet >.

Reprenons l'exemple. On pourrait decrire cela de la façon suivante:

```
DEFINE ENTITY lettre-contact;
  CONSISTS OF ....., support, ..... ;

DEFINE ENTITY demande-inscription;
  CONSISTS OF ....., support, ..... ;

DEFINE ENTITY contacts;
  CONSISTS OF ....., support, ..... ;
```

et définir un seul élément:

```
DEFINE ELEMENT support;
  .....
```

Quels sont les éléments que l'on peut traiter de cette manière ?
Pour le message: contenu, échéance, statut, support.
Pour le document: sujet, corps, échéance, statut, support, paragraphes-statiques.
Pour le formulaire: échéance, statut, support.
Pour le dossier: échéance, support.
Pour le fichier et la pile: support.

Ainsi donc, dans toute transformation, figurera une description d'éléments prédéfinis. Cette description est la suivante:

```
DEFINE ELEMENT contenu;
  FORMAT IS texte;
DEFINE ELEMENT échéance;
  FORMAT IS 99/99/9999;
DEFINE ELEMENT statut;
  DOMAIN OF VALUE ARE original, copie.
DEFINE ELEMENT support;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE ... énumération des supports possibles ...;
DEFINE GROUP sujet;
  CONSISTS OF x mot-clé;
DEFINE ELEMENT mot-clé;
  FORMAT IS alphabétique;
DEFINE ELEMENT corps;
  FORMAT IS texte;
DEFINE ELEMENT paragraphes-statiques;
  FORMAT IS booléen;
```

3.5.2.2 REPRESENTATION EN DSL DU DOCUMENT

Pour ce qui concerne le document, nous allons procéder à une description détaillée et illustrée par des exemples, du processus de transformation. Nous nous baserons sur la transformation du document lettre-contact qui a déjà servi d'exemple dans les développements précédents.

3.5.2.2.1 LA SAISIE DE L'INFORMATION

L'utilisateur doit décrire les objets de bureau selon le modèle de structuration de l'information présenté dans le MIB (chapitre 3). Pour l'aider dans cette tâche, il dispose d'écrans à compléter avec les renseignements demandés.

Pour ce qui concerne le document, l'écran de saisie se présente comme suit:

AJOUT-DOCUMENT.

NOM DU DOCUMENT: lettre-cont
TYPE DU DOCUMENT: lettre
EXPLICATION DU TYPE:

ETATS POSSIBLES:
- reçu-secr
- traité
- classé-inscr
- classé-etd

EXPEDITEUR: étudiant

DESTINATAIRES:
- secrétariat

REPONSES AU DOCUMENT:

NOM	TYPE
lettre-accep	formulaire
dde-inscr	formulaire

3.5.2.2.2 DESCRIPTION DE L'ENREGISTREMENT

L'information fournie par l'utilisateur est vérifiée quant à sa cohérence et à sa complétude: les noms sont-ils compatibles avec DSL ? Tous les renseignements nécessaires ont-ils été fournis ? Leur valeur est-elle correcte ? ... .
Les informations correctes sont rangées en mémoire dans des enregistrements. L'enregistrement correspondant au document se présente comme suit:

```
NOM-UTILISATEUR : ARRAY[1..12] OF CHAR;

DOCUMENT : RECORD
  NOM : NOM-UTILISATEUR;
  TYPE : NOM-UTILISATEUR;
  EXPLICATION : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  EXPEDITEUR : NOM-UTILISATEUR;
  DESTINATAIRES : ARRAY[1..NBRE-DEST] OF NOM-UTILISATEUR;
  REPONSE : ARRAY[1..NBRE-REPONSE] OF NOM-UTILISATEUR;
```

où NOM-UTILISATEUR est un type qui représente des chaines de 12 caractères, c'est-à-dire des noms compatibles avec DSL.

Le document lettre-contact sera donc représenté en mémoire par l'enregistrement:

```
DOCUMENT:
  NOM : lettre-cont
  TYPE : lettre
  EXPLICATION :
  ETAT [1] : reçu-secr
       [2] : traité
       [3] : classé-inscr
       [4] : classé-etd
  EXPEDITEUR : étudiant
  DESTINATAIRES [1] : secrétariat
  REPONSE [1] : lettre-accep
          [2] : dde-inscr
```

3.5.2.2.3 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Pour procéder à la traduction des objets du MIB en DSL, nous faisons appel à un sous-ensemble des concepts de DSL. Les concepts qui nous sont utiles peuvent être rassemblés dans des squelettes de texte DSL selon la solution choisie en 3.4.

Pour le document, étant donné la solution choisie, nous construirons le texte DSL comme suit:

- On donnera le texte décrivant l'objet DSL "ENTITY" représentant le document.
- On donnera le texte décrivant les objets DSL "MESSAGE" représentant le document.
- On donnera le texte décrivant les liens avec les réponses attendues.
- On donnera le texte décrivant les liens avec l'organisation.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation du document à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans le document.

```
DEFINE ENTITY DOCUMENT.NOM_ENT;
  KEYWORD ARE "DOCUMENT";
  CONSISTS OF type-DOCUMENT.NOM,
              identifiant-DOCUMENT.NOM,
              état-DOCUMENT.NOM,
              sujet, corps, échéance, statut,
              support, paragraphes-statiques;
```

On décrit les attributs non prédéfinis au moyen de l'objet DSL "ELEMENT.

```
DEFINE ELEMENT type-DOCUMENT.NOM;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE DOCUMENT.TYPE;

DEFINE ELEMENT identifiant-DOCUMENT.NOM;
  FORMAT IS alphabétique;

DEFINE ELEMENT état-DOCUMENT.NOM;
  DOMAIN OF VALUE ARE DOCUMENT.ETAT[1],
        ...., DOCUMENT.ETAT[NBRE-ETATS];
```

b. Représentation du document à l'aide de l'objet DSL "MESSAGE"

On décrit le document au moyen de l'objet DSL "MESSAGE" afin de permettre à l'information contenue dans le document de circuler dans le système d'information.
Nous avons vu que, si on veut simuler le comportement du système, on doit décrire autant d'objets DSL "MESSAGE" qu'il y a d'états dans le cycle de vie du document.

Pour chaque état du document, on décrira donc:

```
DEFINE MESSAGE DOCUMENT.NOM-DOCUMENT.ETAT[i]_MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "DOCUMENT.NOM_ENT"
  CONSISTS OF type-DOCUMENT.NOM,
              identifiant-DOCUMENT.NOM,
              sujet,
              corps,
              échéance,
              statut,
              support,
              paragraphes-statiques;
```

c. Représentation des réponses attendues.

On représente le lien existant entre l'objet communiqué et les objets attendus en réponse au moyen de l'objet DSL "RELATION".
Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans DOCUMENT.NOM), du nom de l'objet attendu en réponse (contenu dans DOCUMENT.REPONSE) et d'un mot désignant le lien: REP.

Pour chaque réponse attendue, on aura donc:

```
DEFINE RELATION DOCUMENT.NOM_REP_DOCUMENT.REPONSE[i];
  RELATES DOCUMENT.NOM_ENT
    AS a-pour-réponse
    WITH CONNECTIVITY 0-N;
  RELATES DOCUMENT.REPONSE[i]_ENT
    AS répond-à
    WITH CONNECTIVITY 0-N;
```

d. Représentation des liens avec l'organisation

On connait le nom de l'expéditeur et des destinataires.
On connait le nom de l'objet communiqué.
On définit une RELATION entre les objets DSL "ENTITY" représentant la description de l'objet communiqué et celle de l'expéditeur ou de chaque destinataire.

d.1. Représentation de l'expéditeur

Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans DOCUMENT.NOM), du nom de l'expéditeur (contenu dans DOCUMENT.EXPEDITEUR) et d'un mot désignant le lien: EXP.

```
DEFINE RELATION DOCUMENT.NOM_EXP_DOCUMENT.EXPEDITEUR;
  CONSISTS OF date-exp-DOCUMENT.NOM;
  RELATES DOCUMENT.NOM_ENT
    AS expédié-par
    WITH CONNECTIVITY 1-1;
  RELATES DOCUMENT.EXPEDITEUR_ENT
    AS expédie
    WITH CONNECTIVITY 0-N;

DEFINE ELEMENT date-exp-DOCUMENT.NOM;
  FORMAT IS 99/99/9999;
```

d.2. Représentation des destinataires

Le nom de la relation sera formé à partir du nom de l'objet communiqué, du nom du destinataire (contenu dans DOCUMENT.DESTINATAIRES) et d'un mot désignant le lien: DEST.

Pour chaque destinataire, on aura:

```
DEFINE RELATION DOCUMENT.NOM_DEST_DOCUMENT.DESTINATAIRES[i];
  RELATES DOCUMENT.NOM_ENT
    AS destiné-à
    WITH CONNECTIVITY 0-N;
  RELATES DOCUMENT.DESTINATAIRES[i]_ENT
    AS reçoit
    WITH CONNECTIVITY 0-N;
```

Exemple:

Disposant de l'enregistrement décrit en 3.5.2.2.2. et des squelettes de texte DSL, la transformation en DSL du document lettre-contact donnera le texte suivant:

Représentation du document à l'aide de l'objet DSL "ENTITY"

```
DEFINE ENTITY lettre-cont_ENT;
  KEYWORD ARE "DOCUMENT";
  CONSISTS OF type-lettre-cont,
              identifiant-lettre-cont,
              état-lettre-cont,
              sujet, corps, échéance, statut,
              support, paragraphes-statiques;

DEFINE ELEMENT type-lettre-cont;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE lettre;

DEFINE ELEMENT identifiant-lettre-cont;
  FORMAT IS alphabétique;

DEFINE ELEMENT état-lettre-cont;
  DOMAIN OF VALUE ARE reçu-secr, traité,
       classé-inscr, classé-etd;
```

Représentation du document à l'aide de l'objet DSL "MESSAGE"

```
DEFINE MESSAGE lettre-cont-reçu-secr_MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "lettre-cont_ENT"
  CONSISTS OF type-lettre-cont,
              identifiant-lettre-cont,
              sujet,
              corps,
              échéance,
              statut,
              support,
              paragraphes-statiques;

DEFINE MESSAGE lettre-cont-traité_MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "lettre-cont_ENT"
  CONSISTS OF type-lettre-cont,
              identifiant-lettre-cont,
              sujet,
              corps,
              échéance,
              statut,
              support,
              paragraphes-statiques;
```

```
DEFINE MESSAGE lettre-cont-classé-inscr_MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "lettre-cont_ENT"
  CONSISTS OF type-lettre-cont,
              identifiant-lettre-cont,
              sujet,
              corps,
              échéance,
              statut,
              support,
              paragraphes-statiques;

DEFINE MESSAGE lettre-cont-classé-etd_MES;
  KEYWORD ARE "DOCUMENT";
  ATTRIBUTE IS "associé-à"  "lettre-cont_ENT"
  CONSISTS OF type-lettre-cont,
              identifiant-lettre-cont,
              sujet,
              corps,
              échéance,
              statut,
              support, paragraphes-statiques;
```

Représentation des réponses attendues.

```
DEFINE RELATION lettre-cont_REP_lettre-accep;
  RELATES lettre-cont_ENT
    AS a-pour-réponse
    WITH CONNECTIVITY 0-N;
  RELATES lettre-accep_ENT
    AS répond-à
    WITH CONNECTIVITY 0-N;

DEFINE RELATION lettre-cont_REP_dde-inscr;
  RELATES lettre-cont_ENT
    AS a-pour-réponse
    WITH CONNECTIVITY 0-N;
  RELATES dde-inscr_ENT
    AS répond-à
    WITH CONNECTIVITY 0-N;
```

Représentation des liens avec l'organisation

Représentation de l'expéditeur

```
DEFINE RELATION lettre-cont_EXP_étudiant;
  CONSISTS OF date-exp-lettre-cont;
  RELATES lettre-cont_ENT
    AS expédié-par
    WITH CONNECTIVITY 1-1;
  RELATES étudiant_ENT
    AS expédie
    WITH CONNECTIVITY 0-N;

DEFINE ELEMENT date-exp-lettre-cont;
  FORMAT IS 99/99/9999;
```

Représentation des destinataires

```
DEFINE RELATION lettre-cont_DEST_secrétariat;
  RELATES lettre-cont_ENT
    AS destiné-à
    WITH CONNECTIVITY 0-N;
  RELATES secrétariat_ENT
    AS reçoit
    WITH CONNECTIVITY 0-N;
```

3.5.2.3 REPRESENTATION EN DSL DU FICHIER

Le processus de transformation est le même que pour le document. Nous ne décrirons donc plus en détail que l'enregistrement où seront rangés l'information et les squelettes de texte DSL servant à la transformation.

3.5.2.3.1 DESCRIPTION DE L'ENREGISTREMENT

```
FICHIER : RECORD
  NOM : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  CONSTITUANT : NOM-UTILISATEUR;
```

3.5.2.3.2 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Etant donné la solution choisie, on construira le texte DSL comme suit:

- On donnera le texte décrivant l'objet DSL "ENTITY" représentant le fichier.
- On donnera le texte décrivant le lien entre le fichier et son constituant.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation du fichier à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans le fichier.

```
DEFINE  ENTITY  FICHIER.NOM_ENT;
  KEYWORD  ARE  "FICHIER";
  CONSISTS  OF  état-FICHIER.NOM,
                support;

DEFINE  ELEMENT  état-FICHIER.NOM;
  FORMAT IS  alphabétique;
  DOMAIN OF VALUE ARE  FICHIER.ETAT[1],
      ......, FICHIER.ETAT[NBRE-ETATS];
```

b. Représentation du constituant du fichier

On connait le nom du fichier.
On connait le nom de son constituant.
On définit une RELATION entre les objets DSL "ENTITY" représentant la description du fichier et celle de son constituant.
On aura:

```
DEFINE RELATION FICHIER.NOM_CONS_FICHIER.CONSTITUANT;
  RELATES  FICHIER.NOM_ENT
    AS  contient
    WITH CONNECTIVITY  0-N;
  RELATES  FICHIER.CONSTITUANT_ENT
    AS  contenu-dans
    WITH CONNECTIVITY  0-1;
```

3.5.3 LA REALISATION DE LA TRANSFORMATION

Nous ne décrirons pas, dans ce chapitre, le programme d'implémentation de la transformation des objets informationnels du MIB en DSL. Le lecteur intéressé pourra consulter l'annexe 5 à ce sujet. Cette annexe traite:

- du choix des outils utilisés: gestionnaire d'écran, langage de programmation, langage des commandes DSL-SPEC;
- des spécifications complètes du programme;
- des résultats obtenus: rapports documentaires.

3.5.4 LE RETOUR AU MIB

Une fois que les objets informationnels du MIB ont été transformés en DSL, il faut voir si la transformation inverse est possible: à partir des objets décrits en DSL, pourra-t-on restituer à l'utilisateur toute l'infomation qu'il avait décrite ?

Ce retour au modèle initial est indispensable, car, pour l'utilisateur, DSL n'est qu'un outil générique, non compréhensible.

La restitution de l'information selon le MIB à partir de la traduction en DSL est possible: elle est réalisée par l'analyse des fichiers qui contiennent l'information traduite en DSL.

L'analyse détaillée du processus de transformation de DSL vers le MIB est donnée à l'annexe 5. On peut y trouver:

- les principes du processus de transformation;
- le programme de transformation: spécifications et texte.

CONCLUSION

Dans cette conclusion, nous nous proposons de faire une synthèse du travail réalisé dans ce mémoire. Nous procéderons ensuite à une évaluation du choix du système IDA comme système générique d'implémentation. Nous verrons enfin dans quelle mesure nous avons atteint les résultats fixés et nous donnerons quelques directions de recherche.

1 SYNTHESE

L'objectif de ce mémoire était de fournir un environnement sous forme de modèles et d'outils automatisés, favorable à la construction d'un schéma conceptuel qui soit:

- spécifique au bureau;
- communicable;
- complet;
- cohérent;
- conforme aux besoins.

Pour atteindre cet objectif, nous avons commencé par faire une synthèse de la littérature traitant du même sujet, afin d'étudier les techniques et méthodes d'analyse existantes. Cette étude des modèles existants (chapitre 2) a permis de distinguer trois phases dans la conception d'un système d'information de bureau:

- l'analyse des besoins: On étudie le bureau et les fonctions exécutées.
- la spécification des besoins: On donne une description formelle et aussi complète que possible de tous les aspects du travail de bureau.
- l'implémentation du système.

Nous avons alors procédé à la réalisation de chacune de ces phases.

Première étape: l'analyse des besoins. (Chapitre 3)

A partir d'une application décrivant le flux administratif des étudiants de l'institut d'informatique, nous avons analysé les caractéristiques du bureau:

- la structure organisationnelle;
- les informations manipulées;
- les ressources utilisées;
- les traitements qui y sont associés;
- les enchainements entre ces traitements.

Deuxième étape: la spécification formelle. (Chapitre 3)

Nous sommes partis du diamant de Leavitt, synthèse de l'organisation, dont nous avons décomposé chaque pavé. Cette décomposition nous a permis de souligner les aspects particuliers du modèle général d'information de bureau (MIB). Pour chacun de ces aspects, nous avons défini différents modèles particuliers:

- le modèle de structuration de l'information:
  Il permet de décrire la structure organisationnelle du bureau: l'organigramme de l'organisation, les responsabilités et le rôle des différentes personnes au sein de l'organisation.
- le modèle de structuration de l'information:
  Il sert à définir l'information traitée dans le bureau.
- le modèle des ressources:
  Il décrit les besoins en technologie et en personnel dans la réalisation du but du bureau.
- le modèle de structuration des traitements:
  Il permet une décomposition des traitements en différentes tâches et une découpe élémentaire des tâches en opérations primitives.
- le modèle de la dynamique des traitements:
  Il permet de décrire les enchaînements entre tâches et, pour chaque tâche, entre opérations primitives.

Nous avons ensuite énoncé les règles de complétude et de cohérence du modèle, de façon à vérifier qu'une spécification du comportement d'un système d'information de bureau à l'aide du modèle décrit est une bonne spécification, c'est-à-dire:

- complète: il n'existe pas de composants référencés mais non complètement décrits.
- cohérente: il n'existe pas de contradiction dans la description.

Troisième étape: l'implémentation du système. (Chapitre 4)

Nous avons implémenté notre modèle détaillé à partir d'un système plus générique: le système IDA, car:

- ce système existe: on peut donc récupérer toute la technologie et les outils existants;
- le système IDA fournit plusieurs résultats qui correspondent aux objectifs que nous nous sommes fixés (spécification, documentation, simulation);
- notre philosophie de base consiste à utiliser un outil commun générique sur lequel on grefferait différents interfaces de façon à implémenter divers systèmes plus détaillés.

Nous avons ensuite présenté une solution possible de transformation des objets informationnels du MIB en DSL et nous avons réalisé l'implémentation de cette solution.

2 EVALUATION DU CHOIX DE IDA

Le choix du système IDA en tant que système générique pour l'implémentation de notre modèle était-il un bon choix ?

Dans la recherche d'une solution de transformation des objets informationnels du MIB en DSL, nous avons fait appel à seulement trois concepts du langage DSL: l'objet "MESSAGE", l'objet "ENTITY" et l'objet "RELATION". Les objets DSL destinés à représenter l'organisation (objet "INTERFACE") et à associer des objets (objet "SET") ne répondent pas à toutes les contraintes posées par le MIB. Nous nous sommes donc ramenés aux seuls concepts du modèle entité - association, modèle qui permet de tout représenter si on décrit les objets avec les attributs adéquats. Est-il nécessaire alors de faire appel au langage DSL qui semble peu adapté à notre problème ?

Une seconde raison de douter de l'opportunité de l'usage du système IDA est que les rapports documentaires et les rapports du Query System fournis par ce système sont peu lisibles pour un utilisateur qui ne comprend pas le langage DSL: ils sont écrits en phrases DSL et la description des objets informationnels de bureau telle qu'elle est présentée à l'utilisateur dans le MIB est absente. L'utilisateur du MIB étant sensé ignorer la présence du système IDA, ces rapports sont incompréhensibles pour lui et perdent donc de leur intérêt.

Faut-il pour autant rejeter le choix du système IDA ?
Nous pensons qu'il ne faut pas rejeter catégoriquement ce système générique:

- La seule transformation des objets informationnels nous a permis de produire peu de rapports: il existe d'autres rapports graphiques, mais qui font appel à plus d'information, et qui seraient peut-être plus compréhensibles pour un utilisateur du système bureautique ne connaissant pas le langage DSL.

- Il existe un outil important que nous n'avons pas exploité: l'outil de simulation. N'ayant pas traité la dynamique du système, nous n'avons pu en simuler le comportement. Peut-être le système IDA serait-il vraiment utile sur le plan de la dynamique et de ce fait, tolèrerait-on sa moins bonne adaptation sur le plan descriptif ?

3 EVALUATION DES RESULTATS

Nous avions fixé trois types de résultats:

- la spécification:

Le modèle d'information de bureau que nous avons présenté ici met en évidence toutes les caractéristiques du bureau. Dans l'implémentation des objets informationnels du bureau, nous avons voulu offrir à l'utilisateur un interface agréable pour la description du bureau. Ce modèle adapté au bureau et la perspective d'outils de description agréables facilitent et accélèrent la définition et l'analyse des activités de bureau. Le résultat d'aide à la spécification est donc atteint.

- la documentation:

Comme nous en avons déjà discuté dans le point précédent, nous avons pu produire des rapports documentaires correspondant à l'extraction de descriptions et à l'interrogation du système, mais ces rapports, écrits en DSL, sont peu lisibles pour un utilisateur qui ne connait pas ce langage.

- la simulation:

N'ayant pas implémenté la dynamique, nous n'avons pas atteint ce résultat. Cependant, nous avons tenu compte de ce résultat lors de la recherche d'une solution possible d'implémentation (Chapitre 4: 3.4.2.1. La transformation des objets informationnels proprement dits. B. Le niveau dynamique.)

4 LES DIRECTIONS DE RECHERCHE

Si l'élaboration d'un modèle d'information de bureau a été réalisée complètement, il n'en est pas de même de son implémentation. Beaucoup de travail reste à faire dans ce domaine. Nous avons vu que le choix du système IDA devait être remis en question. Si on décide de garder cette solution d'implémentation, il faudra alors réaliser l'implémentation des modèles de structuration et de dynamique des traitements afin de pouvoir simuler le comportement du système. Si on abandonne le système générique IDA, cherchera-t-on un système mieux adapté ou écrira-t-on complètement un système logiciel pour implémenter le modèle ? La question reste à débattre.

BIBLIOGRAPHIE

[BOD] F. BODART, Y. PIGNEUR.
Conception assistée des applications informatiques:
1. Etude d'opportunité et analyse conceptuelle.
MASSON Presses universitaires de Namur 1983.

[BRA] L.C. BRACKER., B.R. KONSINSKI
Computer-aided analysis of office systems.
MIS Quarterly March 1982.

[CRO] W.B. CROFT, L.S. LEFKOWITZ.
Task support in an office system.
University of Massachusetts.
ACM Transactions on Office Information Systems
vol. 2 _ number 3 _ 1984.

[GIB] S. GIBBS, D. TSICHRITZIS.
A data modelling approach for office information systems.
Beta Gamma edited by D. Tsichrtzis.
Technical Report CSRG_150.

[HAM] M.M. HAMMER, A. SIRBU, S.R. SCHOICHET, J.S. KUNIN, J.B. SUTHERLAND, C.L. ZARMER.
Office analysis: Methodology and case studies.
Massachusetts Institute of Technology 1983.

[IDA 1] Atelier logiciel IDA d'aide à la conception de systèmes d'information.
Synthèse de la syntaxe du langage DSL.
F.N.D.P. Namur

[IDA 2] Idem
Manuel de référence: DSL - SPEC.

[IDA 3] Idem
Manuel de référence: DSL - SIM.

[KUN] JAY S. KUNIN.
Analysis and specification of office procedures.
Massachusetts Institute of Technology 1982.

[LES] R. LESUISSE.
Syllabus de bureautique.
F.N.D.P. NAMUR.

[TSI] D. TSICHRITZIS.
Form management.
Communication of the ACM July 1982 _ vol. 25

[ZLO] M.M. ZLOOF.
Office-by-example: a business language that unifies data and word processing and electronic mail.
IBM Syst J Vol. 21 _ number 3 _ 1982.

Facultés Universitaires Notre-Dame de la Paix - NAMUR

Institut d'informatique

SPECIFICATION ET IMPLEMENTATION D'UN MODELE D'INFORMATION DE BUREAU

(ANNEXES)

Mémoire présenté par

Nadine DACHOUFFE

en vue de l'obtention du titre de Licencié et Maitre en Informatique.

Année académique 1985 - 1986

ANNEXE 1: INSCRIPTION DES ETUDIANTS

L'annexe 1 décrit le flux administratif lié à l'inscription des étudiants de l'institut. Cette application a servi de base à la recherche effectuée dans la conception d'un modèle d'information de bureau. Elle est le résultat d'une analyse du bureau selon la méthode OAM.

1 LES ETUDIANTS DE PREMIERE LICENCE

1.1 LA DEMANDE D'INSCRIPTION

Différentes réponses aux demandes pour différentes catégories d'étudiants.

Parmi les étudiants qui arrivent à l'institut, on en distingue de différentes provenances:

- Les étudiants belges:
  - Les candidats en sciences économiques et sociales ou en mathématique des F.N.D.P.
  - Les autres étudiants belges:
    - Les autres candidats.
    - Les gradués en informatique.
    - Les licenciés.
- Les étudiants étrangers.

Ces catégories requièrent des traitements différents quant aux conditions d'admission.

A. LES ETUDIANTS BELGES

a. Les candidats des F.N.D.P.

Ces étudiants passent directement en première licence, sans aucun examen.
Soit ces étudiants introduisent une demande par lettre ou par téléphone ou en se présentant au secrétariat. Dans ce cas, le secrétariat leur envoie une lettre du type GE_ 1 12 et un bulletin de demande d'inscription en deux exemplaires, l'un étant destiné au secrétariat de l'institut, l'autre au secrétariat central.
Soit ces étudiants n'introduisent aucune demande. Dans ce cas, suite à un avis affiché aux valves, les délégués de cours vont chercher les bulletins d'inscription à remplir en deux exemplaires.

b. Les autres étudiants belges

b.1. Les autres candidats

Ces candidats contactent le secrétariat par lettre, par téléphone ou en se présentant, afin de connaitre les modalités d'inscription.

Ces candidats doivent passer un examen d'entrée. Pour cet examen, certaines dispenses sont accordées.

Le secrétariat envoie à ces étudiants une réponse contenant:
- une lettre disant qu'il y a examen le premier lundi de septembre et mentionnant les dispenses éventuelles (lettre GE_ 1 2);
- deux bulletins de demande d'inscription;
- le programme des cours;
- une brochure décrivant l'institut;
- le détail de l'examen d'entrée.

b.2. Les gradués

Ces étudiants contactent le secrétariat par lettre ou par téléphone ou en se présentant, afin de connaitre les modalités d'inscription. On a alors le même traitement que pour les candidats (b.1.), excepté que les gradués n'ont pas de dispense d'épreuve.

b.3. Les licenciés

Ces étudiants contactent le secrétariat par lettre ou par téléphone ou en se présentant. On a alors le même traitement que pour les autres candidats (b.1.).

Au fur et à mesure que les lettres des étudiants demandant pour s'inscrire arrivent, elles sont rangées dans un classeur "demande de renseignement". Une fois que l'étudiant a renvoyé son bulletin de demande d'inscription rempli, on sort sa lettre de ce classeur. Dans ce classeur restent finalement les demandes restées sans suite.

Si des étudiants veulent s'inscrire sans remplir les conditions requises, le secrétariat leur envoie une seule lettre (lettre GE_ 1 2) et transmet leur dossier au centre de rencontre.

Au fur et à mesure que les demandes d'inscription remplies rentrent, la secrétaire les classe dans un fichier, en attente. Différents renseignements sur les étudiants sont repris sur des fiches par provenance. Ces fiches sont mises dans plusieurs sous-fardes, suivant la provenance de l'étudiant.

A partir de juin, le secrétariat reprend les demandes d'inscription pour traiter celles des étudiants devant passer des examens. La secrétaire envoie à ces étudiants une lettre les priant de se présenter aux examens (lettre GE_ 1 10).
Remarque: normalement, ces lettres devraient être envoyées à la reception de la demande d'inscription remplie.

L'acceptation des demandes d'inscription pour les étudiants devant passer un examen est clôturée le 15 août.
Remarque: Les demandes venant plus tard sont encore acceptées, mais on fixe cette date afin d'éviter que trop de demandes arrivent à la dernière minute.

A partir de ce moment, on établit une liste de passage aux examens d'entrée (Examens d'admission à la licence et maîtrise en informatique). Cette liste est distribuée aux différents professeurs et au secrétaire académique qui doit collationner les résultats.

Quand les étudiants ont passé leurs examens, on affiche les résultats aux valves.

Si l'étudiant est non admis:
On sort sa demande d'inscription du fichier où elle était rangée et on la classe dans un autre fichier, unique pour tous les types d'étudiants, en indiquant la mention "non admis" ou "refusé".

Si l'étudiant est admis:
On ouvre un dossier pour cet étudiant, c'est-à-dire une chemise contenant:

- les lettres écrites par l'étudiant;
- une copie des lettres envoyées à l'étudiant par le secrétariat;
- une feuille indiquant les papiers dus à la rentrée.

B. LES ETUDIANTS ETRANGERS

Ces étudiants écrivent au secrétariat. Ils doivent présenter un examen d'entrée à l'ambassade de leur pays en Belgique. Avant cet examen, on doit établir l'équivalent de leur diplôme (cela est fait par le secrétaire académique). Les étrangers n'ont jamais de dispenses aux examens.

Le secrétariat leur envoie la même documentation qu'aux étudiants belges de la section A.b., en même temps que certains documents sociaux concernant:

- le coût des études;
- le financement des études;
- l'aide du service social;
- les démarches à faire.

Une fois que les étudiants ont renvoyé la demande d'inscription, on se met en contact avec l'ambassade de leur pays, en Belgique, à qui on demande s'ils veulent bien surveiller l'examen. Si l'ambassade donne son accord, on envoie les questions. Les professeurs des facultés corrigent les examens et disent si l'étudiant est admis.

Si l'étudiant est admis, on lui ouvre un dossier.

Si des étudiants étrangers veulent s'inscrire sans remplir les conditions requises, on leur envoie une seule lettre (lettre GE_ 1 2) et on transmet leur dossier au secrétariat central.

1.2 L'INSCRIPTION PROPREMENT DITE

Pour tout étudiant admis, la secrétaire a constitué un dossier contenant le courrier déjà échangé entre l'étudiant et le secrétariat, et une feuille indiquant les papiers dus à la rentrée.

A la rentrée académique, la secrétaire affiche aux valves un avis demandant qu'un délégué de chaque classe viennent chercher les formulaires à remplir.

Les dossiers sont constitués. Ils sont classés par ordre alphabétique. Dans les dossiers, la secrétaire met une photocopie des documents reçus, les originaux étant envoyés au secrétariat central. La secrétaire garde également une photo qu'elle rangera dans un album, en mentionnant le nom et le prénom de l'étudiant. A ce moment, on établit une liste de tous les étudiants de première licence, qui sera distribuée à tous les professeurs en même temps qu'une photocopie de l'album.

On ne s'occupe plus du dossier à partir de ce moment, sauf si on doit écrire à l'étudiant ou si l'étudaint écrit (exemple: demande d'attestation d'inscription aux cours, autres renseignements demandés pour prise de contact, ...).

GE-I 12

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Namur, le 14 août 1985

Monsieur Benoît LEFEBVRE
rue Biolley, 37

4800 VERVIERS

Copie

Monsieur,

Nous avons bien reçu votre demande d'inscription à l'Institut d'Informatique et avons le plaisir de vous signaler que vous êtes accepté, moyennant bien entendu réussite de la Licence en science économique et sociale.

Nous vous informons que la rentrée académique a lieu le lundi 16 septembre et que le cours d'Eléments d'architecture, auquel vous êtes tenu d'assister débutera le 9 septembre à 8H30.

Nous vous prions de croire, Monsieur, à l'expression de nos sentiments très distingués.

C. Mossiat-Ippersiel
Secrétaire

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

INSCRIPTIONS INFORMATIQUE
FACULTES UNIVERSITAIRES N.-D. DE LA PAIX
Rue Grandgagnage 21
5000 NAMUR

INSTITUT D'INFORMATIQUE

DEMANDE D'INSCRIPTION

NOM (majuscules) | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Prénoms (souligner le prénom officiel) .....

Lieu de naissance ..... Date de naissance .....

Nationalité ..... Sexe .....

Adresse personnelle en Belgique :

rue ..... n° ..... à ..... / .....

Prénom du père ..... Profession .....

En cas de décès du père, Nom et prénom de la mère .....

Nombre de frères et sœurs en vie ....., actuellement aux études .....

Téléphone parents : indicatif ..... N° ..... C.C.P. n° .....

ETUDES ANTERIEURES

A) Etudes secondaires :

Nom de l'établissement fréquenté .....

Adresse de l'établissement fréquenté .....

Section suivie (exemple : latin-sciences) .....

Régime linguistique .....

Année de fin d'études secondaires : 19.....

Diplôme d'aptitude à l'enseignement supérieur obtenu le .....

b) Etudes supérieures :

Année	Etablissements	Section	Résultat obtenu
19..... - 19.....			
19..... - 19.....			
19..... - 19.....			
19..... - 19.....			
19..... - 19.....			
19..... - 19.....			

CATEGORIE DEMANDEE (mettre une X)

Externat (retour quotidien en famille) ..... ☐

Home jeunes gens (si possible) ..... ☐

Pédagogie jeunes filles (si possible) ..... ☐

Quartier en ville ..... ☐

Signature de l'étudiant .....

Signature des parents ..... Date .....

AVIS DE L'INSTITUT

☐ 1 Admis, sous réserve de confirmation des résultats en fin d'année et production des documents requis

☐ 2 Admis définitivement

☐ 3 A réexaminer après les résultats de fin d'année

☐ 4 A rencontrer (prière de demander rendez-vous)

☐ 5 Refusé. Motif .....

Date ..... Le Directeur .....

DISPENSES AUX EXAMENS D'ENTREE

Sont dispensés de l'épreuve de mathématique :

- les licenciés en Sciences et en Sciences Economiques,
- les ingénieurs civils,
- les candidats en Mathématique, Physique, Ingénieur Civil, et Informatique, les candidats en Sciences Economiques des F.N.D.P.,

Sont dispensés de l'épreuve de programmation :

- les candidats en Informatique,
- les candidats en Sciences Economiques et en Mathématique des F.N.D.P.,

Sont dispensés de l'épreuve socio-économique :

- les candidats et licenciés en Sciences Economiques, les candidats et licenciés en Sciences Mathématiques des F.N.D.P.

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

GE-1 2

Namur, le Θ

Θ
Θ

Θ

Θ,

Nous avons bien reçu votre lettre du Θ.

L'accès aux études de Licence et Maîtrise en Informatique est soumis à la réussite d'un examen d'entrée dont veuillez trouver en annexe le détail. J'ai cependant le plaisir de vous signaler que vous êtes dispensé de l'épreuve de Θ. Cet examen d'entrée aura lieu les 2 et 3 septembre.

Veuillez également trouver ci-joint une brochure décrivant l'Institut et le programme des cours à l'Institut, étant donné votre diplôme de Θ, le programme qui vous concerne est celui du cycle de Θ ans. Vous trouverez aussi deux formulaires de demande d'inscription à nous renvoyer complétés dès que possible.

En vous souhaitant bonne réception de la présente, nous vous prions d'agréer, Θ, l'expression de nos sentiments distingués.

C. Mossiat-Ippersiel
Secrétaire à l'Institut

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

FACULTÉS
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Namur, le 9 août 1985

Copie

Conseil Régional pour la protection sociale
et du Planning Familial
B.P. 12.477 Kinshasa 1
89, avenue Force Publique

KINSHASA/KASA-VUBU
ZAIRE

Messieurs,

Nous avons bien reçu la demande d'inscription de Monsieur MASSONGELE Safula.

Nous ne pouvons cependant pas l'inscrire à la licence et maîtrise en informatique, car il n'a fait qu'une année de programmeur. Il est nécessaire pour entrer à l'Institut d'avoir fait deux ans de candidature ou de graduat.

Je transmets par même courrier le dossier au secrétariat central qui pourra vous transmettre les renseignements nécessaire pour l'inscription en candidature avec option informatique.

Veuillez agréer, Messieurs, l'expression de mes sentiments très distingués.

C. Mossiat-Ippersiel
Secrétaire

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

EXAMEN D'ADMISSION A L'INSTITUT D'INFORMATIQUE

1. BUT

Le but de l'examen d'entrée est de tester l'aptitude du candidat à aborder, sans rencontrer des difficultés insurmontables, les études de Licence et Maîtrise en Informatique.

La connaissance de certaines matières est requise, mais la réflexion sur les questions posées est plus importante que la mémoire ou que la maîtrise d'outils particuliers.

2. MODALITES

Sauf dispense éventuelle, l'examen comporte trois épreuves ; une épreuve de mathématiques, une épreuve de programmation et une épreuve de culture socio-économique.

3. EPREUVE DE MATHEMATIQUES

3.1. Modalités et objectifs

Pour les étudiants candidats au cycle de **trois** ans, l'examen est écrit ; sa durée est de l'ordre de 3 heures. Il porte d'une part sur un contrôle des connaissances mais aussi, et même principalement, sur la compréhension des concepts (interprétation géométrique, application des définitions, principe de la construction des preuves, etc.) et sur les capacités de raisonnement mathématique.

Pour les étudiants candidats au cycle de **deux** ans, l'examen est oral. Plus encore que pour l'examen écrit, il insiste sur les concepts et les capacités de raisonnement.

3.2. Algèbre

3.2.1. **Matière**

L'examen consiste en la résolution d'applications simples dans les domaines suivants :

Théorie des ensembles

Structures algébriques et nombres complexes.
Notion d'espace vectoriel : base, dimension, éléments générateurs, indépendance linéaire, interprétation géométrique dans Rn.

Calcul matriciel

Propriétés des matrices et opérations matricielles (produit, inversion, rang).
Déterminants (calcul et propriétés).
Systèmes linéaires (paramétrisation et discussion).
Applications linéaires (représentation matricielle, noyau et image).
Valeurs et vecteurs propres (définitions et propriétés, mais pas les méthodes de calcul et de diagonalisation).

3.2.2. **Bibliographie suggérée**

On consultera, par exemple :

- J.P. Thiran, **Algèbre linéaire,** Cours donné en Candidatures en Sciences Economiques et Sociales, Option Informatique, F.N.D.P., Namur, (voir spécialement les chapitres 1 sauf § 8, 2, 3, 4 jusqu'au § 4 inclus, 5, 6, 8 § 1 et 2). Un exemplaire de ceux-ci peut être consulté au Secrétariat de l'Institut d'Informatique).

Une autre référence éventuelle est :

- Lesieur et al., **Algèbre linéaire-Géométrie,** Collection U, Armand Collin, Paris (Intéressant par les exercices proposés).

3.3. Analyse

3.3.1. **Matière**

Limite, continuité et dérivabilité dans Rn

Interprétation géométrique, liaison entre la continuité et la dérivabilité, théorèmes de l'Hospital, de Taylor et de Mac-Laurin.

Intégrales indéfinies et définies, simples et doubles

Intégrale comme limite de somme.
Intégration par parties, par changement de variables.
Critères d'intégrabilité (Quotient, racine, Riemann, Abel).
Calcul des intégrales doubles.
Représentation géométrique.

Suites et séries numériques

Critères de convergence

Equations différentielles

Concepts et méthodes de résolution dans le cas linéaire à coefficients constants et à variables séparées.

3.3.2. Bibliographie suggérée

On consultera, par exemple :

- Demidovitch, **Recueil d'exercices et de problèmes d'analyse mathématique,** MIR., Moscou. Les chapitres importants sont :
  ch 1 dans sa totalité ;
  ch 2 dans sa totalité ;
  ch 4 jusqu'à la page 131 ;
  ch 5 jusqu'à la page 168 ;
  ch 6 jusqu'à la page 221 ;
  ch 7 jusqu'à la page 289 ;
  ch 8 jusqu'à la page 359 ;
  ch 9 § 1, 2, 3, 4 et 12.
  Une autre référence éventuelle est :
- Lesieur-Lefèvre, **Mathématique : Analyse,** Collection U, Armand Collin, Paris (Intéressant par les exercices proposés). En particulier, les chapitres 1 à 6, 8, 12, 14, 16 et 19.

3.4. Logique

Dans le cadre d'un problème simple, le récipiendaire devra donner la preuve de sa compréhension des principes de base de la logique et de la construction des preuves (condition nécessaire et suffisante, démonstration par l'absurde et par récurrence, contraposée).

4. EPREUVE DE PROGRAMMATION

4.1. Objectif

L'examen écrit, d'une durée de deux heures, porte essentiellement sur la conception d'algorithmes sans insister particulièrement sur l'aspect langage.

4.2. Matière

L'examen consiste en la rédaction d'un programme ne présentant que des difficultés classiques ; le langage est laissé au choix du récipiendaire. Les concepts supposés connus sont :

- constante, variable simple, tableau et fichier séquentiel ;
- opérations d'assignation, de lecture et d'écriture ;
- structure de contrôle classiques telles que :
  * structure séquentielle,
  * structure alternative et conditionnelle (if then else),
  * structure itérative (for, while, ...) ;
- déclaration et appel de procédure.

4.3. Bibiliographie suggérée

On consultera, par exemple :

- J. Arsac, **Premières leçons de programmation,** F. Nathan, Paris ;
- N. Wirth, **Systematic programming : Introduction,** Prentice Hall, Englewood Cliffs, New Jersey.

A ceux qui seraient intéressés par une étude plus spécifique du langage Pascal, nous suggérons :

- Crozet et Serain, **Le Langage Pascal,** Masson, Paris.

5. EPREUVE DE CULTURE SOCIO-ECONOMIQUE

5.1.Objectif

L'examen fait appel à la capacité de synthèse du récipiendaire, à sa culture générale et à sa culture socio-économique. (Durée = 2H.)

5.2. Matière

L'examen consiste en la rédaction d'une synthèse critique d'une conférence de niveau universitaire portant sur un sujet lié à l'économie et au management des organisations.

5.3. Bibiliographie suggérée

On consultera, par exemple :

- R. Aron, **18 leçons sur la Société industrielle,** Collection Idées, Gallimard, Paris, 1962 ;
- M. Crozier, E. Friedberg, **L'Acteur et le Système,** Editions du Seuil, Paris, 1977.

6. REMARQUES SUR LA CONNAISSANCE SUPPOSEE DES LANGUES

Les étudiants inscrits à l'Institut auront besoin d'une connaissance sérieuse de l'anglais tant écrit que parlé. De plus, les candidats au cycle de trois ans doivent avoir des connaissances de base en néerlandais, car des ateliers de langue sont prévus au programme. Un effort très intense pour se mettre à jour dans ce domaine est, dès lors, très vivement conseillé.

Manuel et cassette d'exercices dans les deux langues sont disponibles au prix de 300 FB par langue.

*
* *

LICENCE ET MAITRISE EN INFORMATIQUE (Programme de 3 ans) Première épreuve

		Premier semestre						Second semestre					
		lundi	mardi	mercr.	jeudi	vendr.	samedi	lundi	mardi	mercr.	jeudi	vendr.	samedi
I.1. *Programme commun*													
1. Philosophie (1re partie) : Crise des sciences et pensée contemporaine J. BERLEUR.	30 h.	—	8.30 10.30	—		—	—	—	—	—	—	—	—
2. Structure des ordinateurs (1re partie)	45 h.	—	10.40 12.40	—	14.30 16.30 (a)	—	—	—	—	—	—	—	—
J. BRUNIN.													
+ Travaux pratiques	15 h.	—	—	—	14.30 16.30 (b)	—	—	—	—	—	—	—	—
3. Systèmes d'exploitation (1re partie) J. RAMAEKERS.	45 h.	—	—	10.40 12.40	—	—	—	—	—	—	—	8.30 10.30 (a)	—
+ Travaux pratiques	15 h.	—	—	—	—	—	—	—	—	—	—	8.30 10.30 (b)	—
4. Théorie des langages (1re partie) : Techniques de compilation . . H. LEROY.	30 h.	—	—	—	—	—	—	—	—	—	14.00 16.00	—	—
5. Fichiers et banques de données (1re partie)	30 h.	—	—	—	—	—	—	—	—	14.00 16.00	—	—	—
J.-L. HAINAUT.													
+ Travaux pratiques	15 h.	—	—	—	—	—	—	—	—	—	10.40 12.40 (b)	—	—

		Premier semestre						Second semestre					
		lundi	mardi	mercr.	jeudi	vendr.	samedi	lundi	mardi	mercr.	jeudi	vendr.	samedi
6. Langage COBOL (1re partie) . .	30 h.	—	—	—	—	14.00 16.00	—	—	—	—	—	—	—
A. VAN LAMSWEERDE.													
+ Travaux pratiques	30 h.	—	—	—	—	—	—	—	—	—	—	14.00 16.00	—
7. Concepts, méthodes et outils de l'analyse fonctionnelle	30 h.	—	—	—	8.30 10.30	—	—	—	—	—	—	—	—
F. BODART.													
+ Travaux pratiques	30 h.	—	—	—	—	8.30 10.30	—	—	—	—	—	—	—
8. Modèles mathématiques de la recherche opérationnelle (1re partie)	30 h.	—	—	—	—	—	—	—	—	—	8.30 10.30	—	—
J. FICHEFET.													
+ Travaux pratiques	20 h.	—	—	—	—	—	—	—	—	—	—	10.40 12.40 (3)	—
9. (●) Compléments de statistique mathématique	30 h.	—	—	—	—	—	—	—	—	10.40 12.40	—	—	—
M. NOIRHOMME-FRAITURE.													
+ Travaux pratiques	15 h.	—	—	—	—	—	—	—	—	—	10.40 12.40 (a)	—	—
9. (⊙) Introduction à la Statistique mathématique	30 h.	—	—	—	—	—	—	—	—	10.40 12.40	—	—	—
F. LOUVEAUX.													
+ Travaux pratiques	15 h.	—	—	—	—	—	—	—	10.40 12.40 (b)	—	—	—	—

LICENCE ET MAITRISE EN INFORMATIQUE : PREMIÈRE ÉPREUVE

(●) Pour les candidats en Sciences Économiques — option informatique des F.N.D.P.
(⊙) Pour les autres étudiants.

(1) NOM + PRENOM	DATE DE NAISSANCE	ADRESSE	TELEPHONE	DIPLOME	UNIVERSITE	REMARQUES

(1) La secrétaire inscrit à cet endroit l'origine des étudiants figurant sur la fiche:

"CANDIDATS", "LICENCIES" ou "GRADUES"

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

GE-I 10

Namur, le Θ

Θ
Θ

Θ

Θ,

Suite à votre demande d'inscription en première Licence et Maîtrise en Informatique à l'Institut d'Informatique, nous vous prions de vous présenter le lundi 2 septembre à Θ à l'Institut, afin de présenter l'examen d'entrée, qui, nous vous le rappelons sera écrit.

Pour information, l'examen se déroulera selon l'horaire suivant :

- 09H00 à 12H00 : épreuve de mathématique,
- 13H30 à 15H30 : épreuve de socio-économie,
- 15H30 à 17H30 : épreuve de programmation.

Nous vous signalons également que les résultats des examens seront proclamés après la délibération qui se tiendra le vendredi 6 septembre à 9 heures (Salle académique, 4ème étage).
Les étudiants sont priés de vérifier personnellement le résultat de leur examen d'entrée.

Les étudiants qui auront réussi cet examen d'entrée sont priés d'apporter au Secrétariat de l'Institut (3ème étage) pour le lundi 9 septembre les documents suivants :

- l'attestation de réussite de leurs études de graduat ou ingéniorat technique,
- le certificat de maturité donnant accès aux études universitaires,
- une attestation de ressources (bourse, répondant, ...) pour les étudiants étrangers.

Ces conditions sont indispensables pour pouvoir établir votre dossier qui doit être transmis à la Commission d'admission. Celle-ci statuera sur votre admission définitive.

Nous vous signalons enfin que la rentrée académique a lieu le lundi 16 septembre et que le cours d'Eléments d'architecture, obligatoire pour tous sauf pour les gradués en informatique, débutera le lundi 9 septembre à 8H30.

Nous vous prions de croire, Θ, à l'expression de nos sentiments très distingués.

C. Mossiat-Ippersiel
Secrétaire

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

EXAMENS D'ADMISSION A LA LICENCE & MAITRISE EN INFORMATIQUE

ANNEE ACADEMIQUE 1985-86

* = Math. oral
(1) = Licencié Math.
(2) = Licencié Sciences Economiques Appliquées

NOM, Prénom	MATH.	PROGRAM.	ECONO.	RESULTATS
BECKAERT, Koenraad	x	x	x	
BELGE, Francis	x	x	x	
BERNARD, Danielle	*	x	x	
BIENFAIT, Bernadette		x	x	
BOLLEN, Pierre	x	x	x	
BONCHEMMAMA, Saïd	x	x	x	
BROUSMICHE, Jean-Pierre		x	x	
CHABOTEAUX, Guy		x	x	
CITTA, Jean-Pierre	x	x	x	
CLAUDY, Thierry	*	x	x	
COLIGE, Alain		x	x	
COLLARD, Jean-Marc	x	x	x	
CRISMER, Paul-Georges	x	x	x	
DAMBIERMONT, Christine	x	x	x	
DESTREBECQ, Bernard		x	x	
DEVEUX, Denis	x	x	x	
DUBISY, Françoise		x	x	
DUTILLIEU, Robert	x	x	x	
EBNDAOUD, Hassan		x	x	
ELOY, Marc	x	x	x	
FAHSSIS, Rachid		x	x	
FLENER, Pierre	x	x	x	
FRASELLE, Marie-Christine	x	x	x	
GALARZA (?)			x	
GALLOY, Olivier			x	
GERARD, Annick		x	x	
GERARD, Philippe (1)		x	x	
GERARD, Philippe (2)		x		

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Namur, le SEPTEMBRE 1983

AVIS AUX ETUDIANTS

DE PREMIERE LICENCE ET MAITRISE (CYCLES DE 2 ET 3 ANS)

AFIN QUE VOTRE ADMISSION AUX COURS SOIT DEFINITIVE, NOUS VOUS DEMANDONS DE BIEN VOULOIR NOUS PERMETTRE DE COMPLETER VOTRE DOSSIER EN NOUS APPORTANT AU SECRETARIAT (SI VOUS NE L'AVEZ DEJA FAIT) LES DOCUMENTS SUIVANTS.

A. POUR TOUS LES ETUDIANTS :

- ☐ LA DEMANDE D'INSCRIPTION EN DOUBLE EXEMPLAIRE (PORTANT LA SIGNATURE DE VOS PARENTS SI VOUS AVEZ MOINS DE 21 ANS);
- ☐ LA FICHE SIGNALETIQUE INDIVIDUELLE;
- ☐ UN CERTIFICAT DE BONNE SANTE REMPLI PAR UN MEDECIN DE VOTRE CHOIX;
- ☐ CINQ PHOTOS (FORMAT CARTE D'IDENTITE) AU VERSO DESQUELLES VOUS INDIQUEREZ VOTRE NOM ET PRENOM;
- ☐ LE BULLETIN D'INSCRIPTION AUX COURS (DOCUMENT OFFICIEL DU MINISTERE);
- ☐ UN CERTIFICAT DE NATIONALITE.

TOUS CES FORMULAIRES PEUVENT ETRE RECLAMES AU SECRETARIAT, MAIS POUR FACILITER LA TACHE DE CELUI-CI, IL SERAIT PREFERABLE QUE VOUS DESIGNIEZ UN DELEGUE DE CLASSE QUI POURRA AINSI SE CHARGER D'EN FAIRE LA DISTRIBUTION PARMI VOUS ET ENSUITE EN FAIRE LA COLLECTE.

B. POUR LES ETUDIANTS SORTANTS D'UN GRADUAT, INGENIORAT TECHNICIEN OU INDUSTRIEL

EN PLUS DES DOCUMENTS PRECITES EN A. :

- ☐ LE CERTIFICAT ATTESTANT LA REUSSITE DE VOS DERNIERES ETUDES;
- ☐ LE CERTIFICAT ATTESTANT VOTRE REUSSITE A L'EPREUVE D'APTITUDE A L'ENSEIGNEMENT SUPERIEUR UNIVERSITAIRE.

C. POUR LES ETUDIANTS AYANT UN DIPLOME UNIVERSITAIRE TERMINAL (CYCLE 2 ANS) :

EN PLUS DES DOCUMENTS PRECITES EN A. :

- ☐ LE CERTIFICAT DE REUSSITE DES ETUDES ANTERIEURES.

D. POUR LES ETUDIANTS ETRANGERS :

EN PLUS DES DOCUMENTS PRECITES EN A. :

- ☐ PROGRAMME DETAILLE DES COURS DES ETUDES ANTERIEURES;
- ☐ L'EQUIVALENCE DE VOTRE DIPLOME AVEC UN DIPLOME BELGE (S'ADRESSER AU MIN.EDUC.D
- ☐ PHOTOCOPIE CONFORME DE VOTRE DIPLOME FINAL PRECEDANT;
- ☐ DIPLOME D'APTITUDE A L'ENSEIGNEMENT SUPERIEUR UNIVERSITAIRE (EX.BACCALAUREAT);
- ☐ UNE ATTESTATION DE RESSOURCE (BOURSE, REPONDANT, REVENUS PROPRES, ...)
- ☐ UN CERTIFICAT DE NATIONALITE;
- ☐ UNE ASSURANCE MALADIE.

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

ANNEE ACADEMIQUE 19../19..

FACULTES UNIVERSITAIRES
Notre-Dame de la Paix
Namur

INSTITUT D'INFORMATIQUE

Etudes suivies : ☐ Licence et Maîtrise (cycle de 3 ans)
☐ Licence et Maîtrise (cycle de 2 ans)
☐ Elève libre

Photo

NOM (majuscules): .....
Prénoms :.....
Lieu et date de naissance :.....
Nationalité :.....
Sexe :.....
Adresse des parents : .....
.....
N° de Tél. : ...../.....
Adresse de l'étudiant : .....
.....
N° de Tél. : ...../.....

C.C.P. ou Compte bancaire : .....

ETUDES ANTERIEURES

a) Etudes secondaires : - Nom de l'établissement fréquenté : .....
.....
- Adresse de l'établissement fréquenté : .....
.....
- Section suivie (Ex. Latin-Sciences) : .....
- Régime linguistique :.....
- Année de fin d'études secondaires : .....
- Diplôme d'aptitude à l'enseignement supérieur obtenu le : .....

b) Etudes supérieures :

ANNEE	Etablissement frequenté + adresse	Section	Résultats
19.. - 19..	.....	.....	.....
19.. - 19..	.....	.....	.....
19.. - 19..	.....	.....	.....
19.. - 19..	.....	.....	.....
19.. - 19..	.....	.....	.....
19.. - 19..	.....	.....	.....

Date : le .../.../19..

Signature :

Date de l'obtention du dernier diplôme (ex. gradué, candidat en ..., licencié) : .....
.....
Intitulé exact du diplôme : .....
.....
.....

Cachet de l'Etablissement

N° du Rôle — 6 à 9

2e inscrip. — 78 (en cas d'inscription complémentaire tracer un x)

N — 5

Inscrire les 2 derniers chiffres du premier millésime de l'année académique : — 1 2

3 4

Cases à compléter par l'Etablissement

10

60 61 62

11 12 13 14 15 16

17 18 19

20 63

64 65 66

21

22 23

67 68

69 70 71 72

24 25 26 27

28

29

30 (1)

31 32

33

34 35 36 37

38 39

40 41 42 43

Bulletin d'Inscription aux Cours

NOM : *(en majuscules)* — 44 à 59

SEXE : Masculin / Féminin (**)

PRÉNOMS : *(dans l'ordre figurant sur la carte d'identité)*

Etat civil : Célib. / Marié / Divorcé / Veuf (**)

LIEU DE NAISSANCE : ..... PAYS : .....

DATE DE NAISSANCE : .....

NATIONALITÉ : ..... CARTE D'IDENTITÉ N° / PASSEPORT N° .....

ADRESSE DE L'ÉTUDIANT : rue ..... n° .....
à ..... Tél. ..... (*)

ADRESSE DES PARENTS : (OU DOMICILE LÉGAL DE L'ÉTUDIANT) rue ..... n° .....
à ..... Prov. .....
(indiquer la commune et non pas le canton postal)

Prénom du père : ..... Tél. : ..... C.C.P. : .....

Nombre de frères et sœurs en vie : .....

A QUEL TITRE AVEZ-VOUS ÉTÉ ADMIS A L'ENSEIGNEMENT UNIVERSITAIRE EN BELGIQUE?

— EN PRODUISANT un certificat BELGE de fin d'études secondaires (humanités, enseignement technique ou artistique reconnu équivalent) ? OUI / NON — Homologué ou en voie d'homologation / Non homologué (**)

Si oui, de quelle section ? ..... Obtenu en 19 .....

De quelle école ? ..... Français / Néerlandais (**)

Avez-vous réussi l'examen de maturité en vue de l'obtention du diplôme d'aptitude à accéder à l'enseignement supérieur ? OUI / NON (**) (***)

— EN PRODUISANT un autre titre (autres titres de fin d'études, diplôme d'examen d'admission, équivalence) ?

Si oui, lequel ? .....

INSCRIPTION PRISE : (Remplissez un bulletin par inscription et par année d'études).

.....

.....

Dans quel établissement d'ENSEIGNEMENT UNIVERSITAIRE en Belgique avez-vous commencé vos études ?

.....

En quelle année ? .....

Les avez-vous interrompues ? Si oui, pendant combien d'années ? .....

Quelles études universitaires poursuiviez-vous l'année dernière ou en dernier lieu ? .....
.....

Dans quel établissement ? .....

Possédez-vous déjà un diplôme d'ENSEIGNEMENT UNIVERSITAIRE ? .....

Si oui, lequel ? .....

Signature :

Date : .....

(*) L'étudiant est tenu de faire connaître au Secrétariat tout changement d'adresse.
(**) Biffer les mentions inutiles.
(***) Cette question ne concerne pas les étudiants sortis des humanités avant 1965, à moins qu'ils ne désirent bénéficier des dispositions de la loi du 8 juin 1964.

Je soussigné.....

Docteur en Médecine O.M. N°.....

Certifie que Mr. Mlle.....

est apte aux études universitaires et à la pratique normale des sports.

Fait à ....., le.....

Cachet Signature

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Namur, le Θ

Θ
Θ
Θ
Θ

Θ

Θ,

Nous avons bien reçu votre lettre du Θ.

L'accès aux études de Licence et Maîtrise en Informatique est soumis à la réussite d'un examen d'entrée qui doit se passer à l'ambassade de Belgique de votre pays avant le mois de juillet. Après avoir reçu votre demande d'inscription, nous prendrons contact avec l'ambassade afin que se déroule cet examen, dont veuillez trouver en annexe le détail.

Veuillez également trouver ci-joint une brochure décrivant l'Institut et le programme des cours à l'Institut, étant donné votre futur diplôme Θ, le programme qui vous concerne est celui du cycle de Θ ans. Vous trouverez aussi deux formulaires de demande d'inscription à nous renvoyer complétés dès que possible.

Il est cependant indispensable pour votre inscription d'être en possession d'une équivalence de diplôme, ainsi que d'un certificat de maturité donnant accès aux études universitaires. Pour obtenir ces documents, vous devez formuler une demande écrite, documents à l'appui, de la manière suivante :

1. une lettre précisant l'objet et le motif de la demande ;
2. un certificat de nationalité ;
3. une copie certifiée conforme à l'original du certificat ou du diplôme d'enseignement secondaire obtenu à l'étranger ou du certificat de réussite de l'épreuve donnant accès aux études supérieures accomplies à l'étranger (ex. Baccalauréat) ;
4. une copie certifiée conforme à l'original du certificat ou du diplôme d'enseignement supérieur obtenu, accompagnée, le cas échéant, d'une traduction en langue française certifiée par un traducteur juré ;
5. le programme officiel détaillé, année par année, des études supérieures accomplies à l'étranger, accompagné éventuellement d'une traduction en langue française ;
6. les notes obtenues aux différents examens sanctionnant ces études, attestées par le chef de l'établissement ;

à l'adresse suivante :

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

Ministère de l'Education Nationale et de la Culture Française
Direction générale de l'Enseignement Supérieur et de la recherche scientifique
Cité Administrative de l'Etat
Bloc Arcades D, 6ème étage
Bureau 6530
A l'attention de Monsieur LAERMANS
rue Royale, 204
1010 BRUXELLES

Il nous est également nécessaire d'avoir une attestation de ressources, celle-ci pouvant être soit une attestation de bourse, soit une attestation de prise en charge signée par vos parents ou toute autre personne qui supporterait effectivement le coût de vos études, soit une attestation faite par vous-même dans laquelle vous certifiez disposer de ressources suffisantes. Pour votre information, nous joignons à la présente une lettre du Service Social relative aux dépenses annuelles que vous serez amené à faire, ainsi qu'un plan de financement que le Service Social demande de bien vouloir lui retourne complété et signé au plus tôt.

En vous souhaitant bonne réception de la présente, nous vous prions d'agréer, Θ, l'expression de nos sentiments distingués.

C. Mossiat-Ippersiel
Secrétaire à l'Institut

P.S.
Nous attirons cependant votre toute spéciale attention sur le fait que depuis septembre 82, les étudiants étrangers doivent payer un important minerval (plus ou moins 141.000 F), sauf quelques rares exceptions.

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX

NAMUR

SERVICES SOCIAUX

RUE DE BRUXELLES, 61
B - 5000 NAMUR
TEL. : (081) 22.90.61
TELEX : 59.222 FAC. NAM. B

Namur, le 28 février 1986

Mademoiselle, Monsieur,

Vous avez sollicité votre inscription aux Facultés. Nous souhaitons quant à nous, si vous êtes admis à vous inscrire, que votre séjour en Belgique soit le meilleur possible. C'est pourquoi nous avons pensé qu'il serait utile de vous donner quelques informations sur le coût d'une année d'études dans notre pays.

Cela vous aidera à prévoir les ressources qui vous seront nécessaires pour vivre dans des conditions favorables à l'étude. Vous devez vous assurer de pouvoir disposer de ces ressources avant même de prendre la décision de venir étudier en Belgique.

Les dépenses annuelles (12 mois) d'un étudiant étranger assimilé aux étudiants belges ou ressortissant d'un pays en voie de développement s'élèvent approximativement à 220 000 FB se décomposant comme suit :

- 11 350 FB (pouvant se ramener à 5 350 FB pour ceux qui bénéficient d'une réduction de minerval) pour l'inscription à l'Université, y compris une session d'examens;

- 54 000 FB (+ un mois de garantie locative) pour le logement en ville;
  ou 30 500 FB pour le logement au home (garçons) et 33 300 FB pour le logement à la pédagogie (filles)

- 70 000 FB pour l'alimentation (somme calculée sur le tarif réduit appliqué par le restaurant universitaire

- 15 000 à 20 000 FB pour les livres, cours, matériel didactique et papeterie;

- 4 000 FB pour les cotisations à l'assurance soins de santé qui est obligatoire;

- 15 000 FB pour l'habillement la première année (éventuellement moins par la suite);

- 15 000 FB pour les loisirs;

- 30 000 FB pour les frais divers (téléphone, médicaments, déplacements, teinturerie, draps de lit la première année, etc.);

Nous attirons votre attention sur le fait qu'une partie importante de votre budget devra être débloquée dans les trois premiers mois : les frais d'inscription, la garantie locative si vous ne logez pas au home ou à la pédagogie, les livres, cours, matériel didactique et papeterie, les draps de lit, c'est-à-dire environ 30 000 FB en plus de ce qu'il vous faudra ordinairement par mois pour vous loger, vous nourrir, vous vêtir et assumer vos frais divers.

Comme vous le constatez, il est indispensable que vous vous assuriez dès maintenant d'avoir les ressources nécessaires pour toute une année d'études car la charge est lourde et il est illusoire de croire que la situation pourra s'arranger en Belgique si elle n'est pas favorable dès le départ.

Nous restons à votre entière disposition pour toute précision ou renseignement complémentaire qui vous serait utile en vue de votre venue éventuelle en Belgique.

Si vous estimez pouvoir réaliser votre projet et si vous êtes admis aux Facultés, nous aurons l'occasion de vous transmettre d'autres informations relatives aux démarches que vous devrez entreprendre.

Nous vous prions d'agréer, Mademoiselle, Monsieur, l'expression de nos sentiments les meilleurs

Le Service Social des Etudiants,

A.-M. DRUET
M.-C. JACQUES
I. MERTENS

CONFIDENTIEL

PLAN DE FINANCEMENT DES ETUDES

Année académique 19.. - 19..

Nom (en lettres majuscules) : .....
Prénom : ..... Nationalité : .....
Lieu et date de naissance : .....
Adresse actuelle : .....
.....
Etat civil : .....
Pour les étudiants mariés : date du mariage : .....
nombre d'enfants : .....
Père : nom et prénom : .....
profession : .....
adresse : .....
.....
Mère : nom et prénom : .....
profession : .....
adresse : .....
.....
Pour les étudiants mariés :
Conjoint : nom et prénom : .....
profession : .....
adresse : .....
.....
Quelles études désirez-vous entreprendre aux Facultés ?
.....
Comment comptez-vous assurer le financement de vos études ?
a) Par une bourse : ... ; si oui, de quel organisme ? .....
Quel en est le montant mensuel ? .....
b) Par un membre de votre famille : ... ; si oui, précisez son identité et le lieu de sa résidence : .....
.....
Quel est le montant mensuel de l'aide ? .....

c) Par un tiers : ... ; si oui, précisez son identité et le lieu de sa résidence : .....
.....
Quel est le montant mensuel de l'aide ? .....

d) Par votre travail ou celui de votre conjoint : ... ; si oui, quel est le salaire mensuel ? .....

e) Par un prêt : ... ; si oui, de quel organisme ? .....
Quel est le montant mensuel du prêt ? .....

f) Par un autre moyen : ... ; si oui, précisez-en la nature et le montant mensuel : .....

Date :
Signature :

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

Service social Etudiants
RUE DE BRUXELLES, 61
B - 5000 NAMUR
TEL. 081/2290.61

Namur, le date de la poste

Mademoiselle,
Monsieur,

Le Service social des Etudiants est heureux de vous accueillir et espère que votre séjour en Belgique vous apportera de nombreuses satisfactions tant sur les plans scientifique et culturel que sur le plan humain.

Dès à présent, nous souhaitons vous inviter à une séance d'information que nous organisons peu après la rentrée académique dans le but de faciliter au maximum votre adaptation à la Belgique. Vous serez averti(e) des lieu et date de cette rencontre par des affiches et une annonce dans les auditoires.

Nous joignons à cette lettre un résumé des démarches à entreprendre avant et après votre arrivée en Belgique.

Dès maintenant, le Service social est à votre entière dispòsition pour tout renseignement, pour toute explication et pour toute aide que vous souhaiteriez obtenir pour vous établir et séjourner à Namur.

En attendant le plaisir de vous voir, nous vous prions d'agréer, Mademoiselle, Monsieur, l'expression de nos sentiments les meilleurs.

Le Service social des Etudiants

RENSEIGNEMENTS PRATIQUES DESTINES AUX ETUDIANTS ETRANGERS ADMIS AUX FACULTES N.D. DE LA PAIX

I. DEMARCHES A ENTREPRENDRE AVANT VOTRE ARRIVEE EN BELGIQUE

1. Régularisation de séjour

Vous devez obtenir une autorisation de séjour provisoire.
Cette attestation doit être demandée auprès du poste diplomatique ou consulaire belge dans votre pays.
Le poste diplomatique ou consulaire belge vous délivrera votre visa d'entrée pour études si vous produisez les documents suivants :

a. une attestation d'admission aux Facultés (délivrée par les Facultés) ;
b. une attestation de ressources mentionnant un montant mensuel minimum de 12.000 FB (par exemple une attestation officielle de bourse);
c. une attestation de prise en charge.
   Cette attestation doit être établie par une personne, belge ou étrangère, qui dispose de ressources suffisantes et s'engage à l'égard de l'Etat belge et de l'étudiant, à prendre en charge les soins de santé, les frais de séjour, d'études et de rapatriement de l'étranger pour au moins une année académique.
d. un certificat médical attestant l'absence de toute maladie pouvant mettre en danger la santé publique, l'ordre public ou la sécurité publique ;
e. un certificat de bonne vie et moeurs pour les étudiants de plus de 21 ans.

ATTENTION : votre visa d'études est indispensable pour pouvoir régulariser votre séjour lors de votre arrivée en Belgique. Les étudiants qui arriveraient en Belgique avec un visa touristique ne peuvent séjourner dans le pays plus de trois mois sans s'exposer à de très sérieuses difficultés administratives.

2. Demande de logement

Il serait prudent que vous vous préoccupiez dès à présent de votre logement.
Deux possibilités s'offrent à vous :

a. ou bien vous réservez une chambre au home (garçons) ou à la pédagogie (filles) en joignant l'avis d'acceptation qui vous a été délivré par le Secrétariat central des Facultés.
   Coût : 27.000 FB par an pour le home, 30.000 FB pour la pédagogie ;
b. ou bien vous prendrez un logement en ville. Coût : environ 48.000 FB par an. Dans ce cas, adressez-vous au service des Relations publiques dès votre arrivée à Namur.

II. DEMARCHES A ENTREPRENDRE DES VOTRE ARRIVEE EN BELGIQUE

1. Vous êtes attendu au Service social des Etudiants qui est à votre disposition si vous le désirez ;

2. Inscription au Registre de population

Dans les huit jours suivant votre arrivée en Belgique, vous êtes tenu de vous présenter à l'Administration communale (Service de population) en étant muni :
- de votre passeport avec visa d'études ;
- de trois photos d'identité.
Vous y obtiendrez votre titre de séjour, document qui constitue la preuve que vous résidez régulièrement en Belgique.

3. Inscription à la mutuelle

En Belgique, la population est obligatoirement assurée contre les soins de santé. Pendant votre séjour dans notre pays, vous devez également être assuré dans ce domaine.
Pour cela, vous devez vous inscrire à une mutuelle de votre choix et payer une cotisation mensuelle. Cela vous permettra, après un stage de 6 mois, d'être remboursé de la plus grande partie de vos frais médicaux et pharmaceutiques.
Le Service social vous donnera les renseignements relatifs aux mutuelles dès votre arrivée à Namur.

4. Nous vous informons également que le Service des Relations publiques tient à votre disposition une brochure qui vous donnera un aperçu général de la vie universitaire au sein de la ville. Y est joint un plan de Namur qui vous aidera dans vos déplacements. C'est là que vous pourrez consulter la liste des logements.

III. LISTE DES ADRESSES UTILES

a. Service social des Etudiants
Centre social
rue Bruno
5000 Namur

b. Service des Logements
Centre social
rue Bruno
5000 Namur

c. Service des Relations publiques
53, rue de Bruxelles
5000 Namur

d. Administration communale
35, rue de Fer
5000 Namur

ANNEXE 2: DEFINITIONS DSL DES CONCEPTS UTILISES

Dans cette annexe, nous allons présenter un rappel des définitions dans le langage DSL des concepts utilisés pour la description des objets du modèle proposé en DSL. Ces définitions sont tirées de [IDA 1].

1 DEFINITION DE L'OBJET "ENTITY"

```
DEFINE ENTITY ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    COLLECTED IN ( SET-name : ,) ;

    CONSISTS OF ( [ { SYSTEM-PARAMETER-name | < integer > } ]
                            { GROUP-name | ELEMENT-name } : , );

    IDENTIFIED BY { GROUP-name | ELEMENT-name | ROLE-name |
                    AGGREGATE-name } ;

    KEYWORDS ARE ( < string > : , ) ;

    RELATED BY RELATION-name
      [ AS ROLE-name ]
      [ WITH CONNECTIVITY { SYSTEM-PARAMETER-name | <string> } ] ;

    [ { STRONG | WEAK } ] SUBTYPES ARE ( ENTITY-name : , ) ;

    [ { STRONG | WEAK } ] SUBTYPE OF ENTITY-name ;
```

2 DEFINITION DE L'OBJET "MESSAGE"

```
DEFINE MESSAGE ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    COLLECTED IN ( SET-name : ,) ;

    CONSISTS OF ( [ { SYSTEM-PARAMETER-name | < integer > } ]
                            { GROUP-name | ELEMENT-name } : , );

    ON GENERATION CONTRIBUTES TO
              [ { ONE | MANY } ] SYNCH-POINT-NAME;

    ON GENERATION TRIGGERS ( PROCESS-name : , );

    KEYWORDS ARE ( < string > : , ) ;

    INCLUDES CONTENT OF MESSAGE-name : , ) ;

    SAME DATA-STRUCTURE AS MESSAGE-name ;

    SUBSETS ARE ( MESSAGE-name : , ) ;

    SUBSET OF ( MESSAGE-name : , ) ;
```

3 DEFINITION DE L'OBJET "ELEMENT"

```
DEFINE ELEMENT ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    DOMAIN OF VALUE ARE ( { SYSTEM-PARAMETER-name | <any-value> }
        [ THRU { SYSTEM-PARAMETER-name | <any-value> } ] : , ) ;

    FORMAT IS <string> ;

    IDENTIFIES { ENTITY-name | RELATION-name } ;

    SAME DOMAIN AS ELEMENT-name ;
```

4 DEFINITION DE L'OBJET "GROUP"

```
DEFINE GROUP ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    CONSISTS OF ( [ { SYSTEM-PARAMETER-name | <integer> } ]
                  { GROUP-name | ELEMENT-name } : , );

    IDENTIFIES { ENTITY-name | RELATION-name } ;

    KEYWORDS ARE ( <string> : , ) ;
```

5 DEFINITION DE L'OBJET RELATION

```
DEFINE RELATION ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    COLLECTED IN ( SET-name : , ) ;

    CONSISTS OF ( [ { SYSTEM-PARAMETER-name | <integer> } ]
                        { GROUP-name | ELEMENT-name } : , );

    IDENTIFIED BY { GROUP-name | ELEMENT-name | ROLE-name
                    | AGGREGATE-name } ;

    KEYWORDS ARE ( <string> : , ) ;

    RELATES ENTITY-name
     [ AS ROLE-name ]
     [ WITH CONNECTIVITY { SYSTEM-PARAMETER-name | <string> } ] ;
```

6 DEFINITION DE L'OBJET "ROLE"

```
DEFINE ROLE ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    IDENTIFIES { ENTITY-name | RELATION-name } ;

    KEYWORDS ARE ( <string> : , ) ;

    ASSUMED BY ENTITY-name IN RELATION-name
     [ WITH CONNECTIVITY { SYSTEM-PARAMETER-name | <string> } ] ;
```

7 DEFINITION DE L'OBJET "SET"

```
DEFINE SET ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    COLLECTION OF ( { ENTITY-name | RELATION-name
                      | MESSAGE-name } : , ) ;

    KEYWORDS ARE ( <string> : , ) ;

    SUBSETS ARE ( SET-name : , ) ;

    SUBSET OF ( SET-name : , ) ;
```

8 DEFINITION DE L'OBJET "INTERFACE"

```
DEFINE INTERFACE ( NAME : , ) ;

    SYNONYM ( synonym : , );

    DESCRIPTION ;
       text;

    ATTRIBUTE IS ( ATTRIBUTE-name { SYSTEM-PARAMETER-name
                                    | < any-value > } : , ) ;

    COLLECTION OF ( { ENTITY-name | RELATION-name
                      | MESSAGE-name } : , ) ;

    GENERATES [ { SYSTEM-PARAMETER-name | <integer> } ]
              MESSAGE-name [ IF CONDITION-name ] ;

    GENERATES [ { SYSTEM-PARAMETER-name | <integer> } ]
              MESSAGE-name IF NOT CONDITION-name;

    KEYWORDS ARE ( <string> : , ) ;

    SUBPARTS ARE ( INTERFACE-name : , ) ;

    PART OF INTERFACE-name ;

    RECEIVES ( [ { SYSTEM-PARAMETER-name | <integer> } ]
               MESSAGE-name : , ) ;
```

ANNEXE 3: LA TRANSFORMATION DES OBJETS INFORMATIONNELS EN DSL

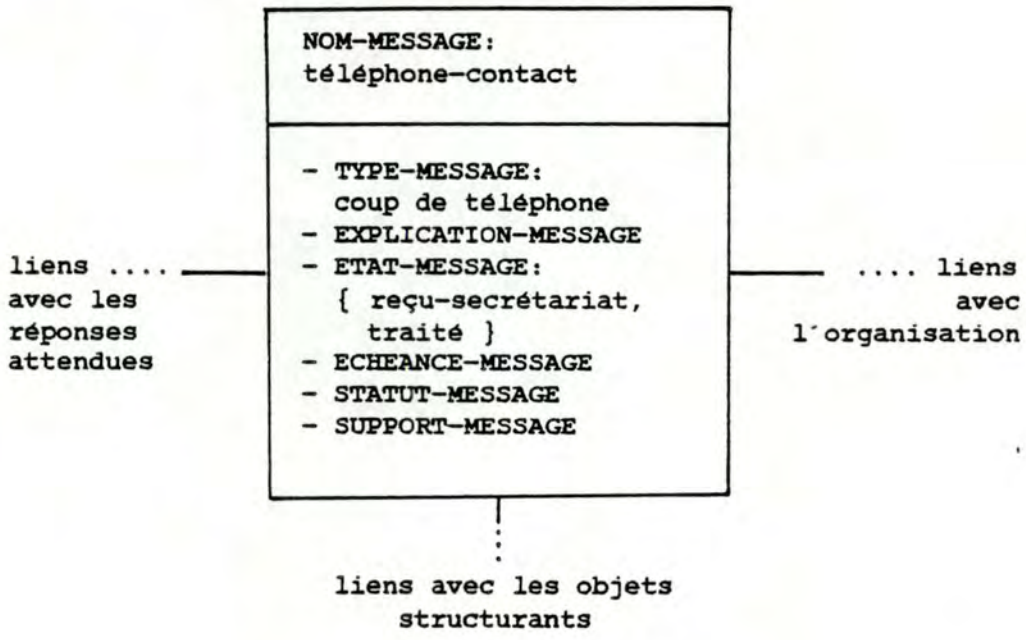
Dans le chapitre 4, nous avons décrit la transformation, en DSL, des objets de bureau "document" et "fichier". Dans cette annexe, nous allons décrire la transformation des autres objets: message, formulaire, dossier et pile. Nous procèderons à cette transformation en se basant sur la solution finalement retenue, c'est-à-dire "A un objet du modèle proposé correspond un objet DSL "ENTITY" et autant d'objets DSL "MESSAGE" qu'il y a d'états dans son cycle de vie."

1 DESCRIPTION DU MESSAGE

Prenons à titre d'exemple le message téléphone-contact.

1.1 LE MESSAGE DANS LE MODELE PROPOSE

Dans le modèle proposé, on avait:



1.2 LE MESSAGE EN DSL

On va décrire d'abord l'information contenue dans l'objet de bureau "message" au moyen de l'objet DSL "ENTITY":

- On peut définir globalement l'objet du MIB "message" au moyen de la clause DEFINE ENTITY.
  Exemple: DEFINE ENTITY téléphone-contact.
- Chacun des attributs du message du modèle proposé sera décrit au moyen de la clause CONSISTS OF.
  L'attribut EXPLICATION-TYPE est facultatif; il donne le type du message quand il ne s'agit pas d'un type standard. Cette explication apparait comme un commentaire. C'est pourquoi, on préfèrera la décrire au moyen de la clause DESCRIPTION.
  Exemple:

```
DEFINE ENTITY  téléphone-contact;
  DESCRIPTION;
      explication-type;
  CONSISTS OF  type-message;
               explication-message,
               état-message,
               échéance-message,
               statut-message,
               support-message;
```

- Pour la description proprement dite des attributs, on fera appel aux objets DSL "ELEMENT" et "GROUP".
  - On définira globalement les attributs au moyen des clauses DEFINE ELEMENT et DEFINE GROUP;
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.

  Exemple:
  pour l'attribut type-message, on aura:

```
DEFINE ELEMENT type-message;
  DOMAIN OF VALUE ARE  coup-de-téléphone;
```

- Sachant que, dans les résultats, on voudrait pouvoir obtenir une liste de tous les messages décrits, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type de l'objet décrit.
  Exemple:

```
DEFINE ENTITY téléphone-contact;
  KEYWORD ARE "MESSAGE";
```

Après avoir décrit l'information contenue dans l'objet "message" du MIB au moyen de l'objet DSL "ENTITY", on le représente par autant d'objets DSL "MESSAGE" qu'il y a d'états dans son cycle de vie.
Pour chaque état:

- On définira globalement le message du MIB au moyen de la clause DEFINE MESSAGE.
  Exemple:

```
DEFINE MESSAGE téléphone-contact-reçu-secrétariat;
DEFINE MESSAGE téléphone-contact-traité;
```

- Les attributs seront les mêmes que pour l'objet DSL "ENTITY". On les décrira au moyen de la clause CONSISTS OF sans les redécrire de façon individuelle.
- Dans la clause KEYWORD, on décrira encore le type de l'objet.
- Afin de faire le lien entre la représentation du message du MIB à l'aide de l'objet DSL "MESSAGE" et sa représentation à l'aide de l'objet DSL "ENTITY", on utilisera la clause ATTRIBUTE IS pour définir l'attribut "est-associé-à".

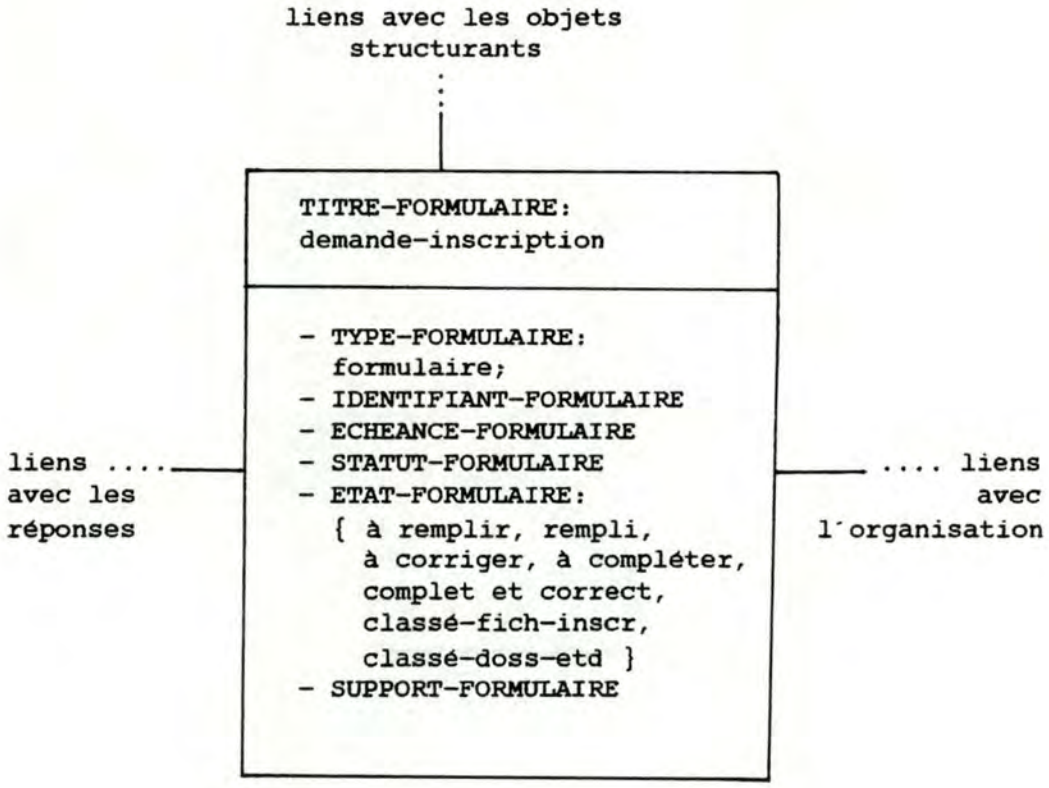
On peut trouver un exemple de transformation d'un message du modèle proposé en DSL à l'annexe 5 (Implémentation: Les rapports de la commandes IP: A.1).

2 DESCRIPTION DU FORMULAIRE

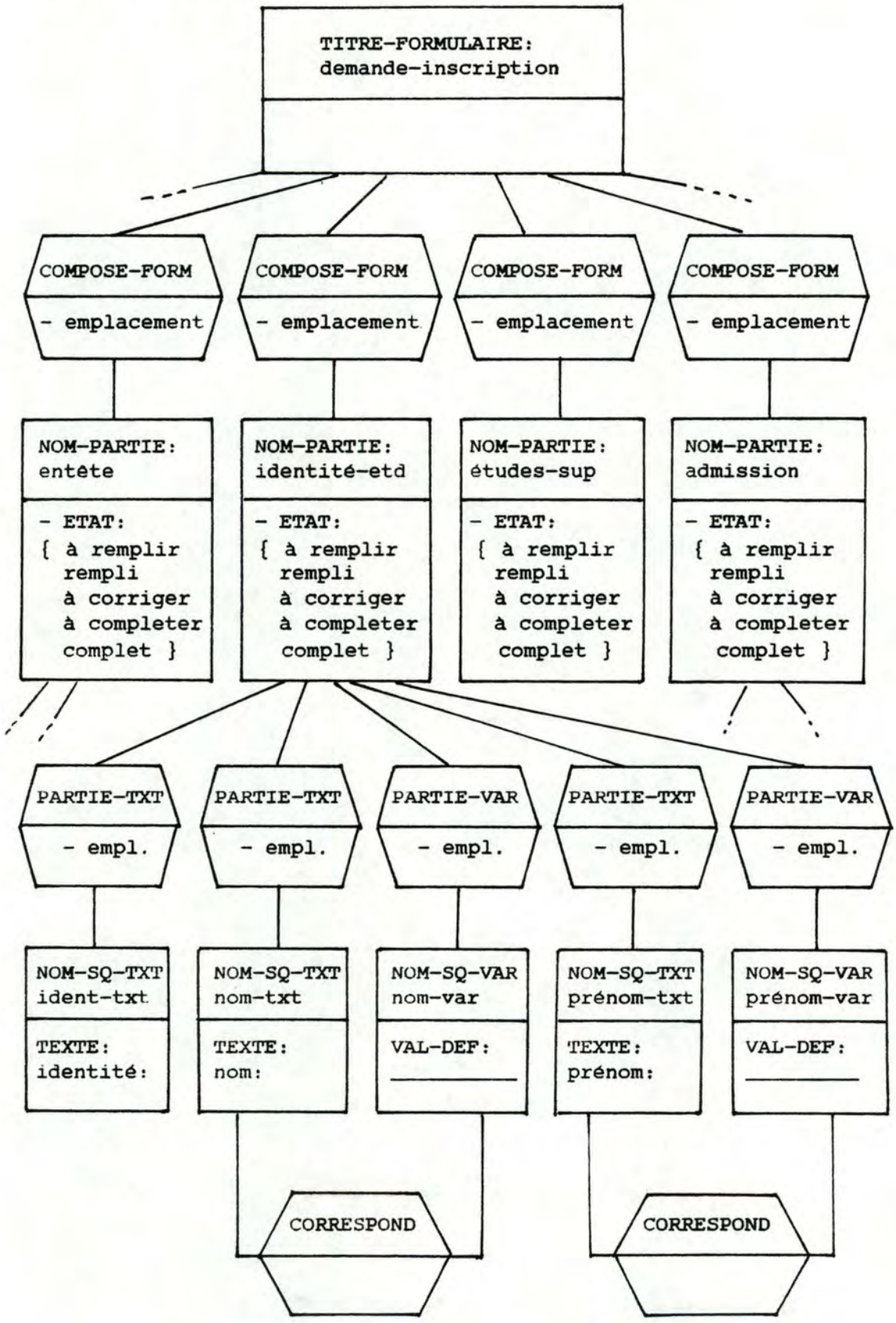
Prenons comme exemple le formulaire demande-inscription.

2.1 LE FORMULAIRE DANS LE MODELE PROPOSE

Dans le modèle proposé, on avait:



et on avait représenté son contenu par:



2.2 LE FORMULAIRE EN DSL

Le détail du contenu du formulaire rend sa description plus complexe que celle du message ou du document. Mais les problèmes fondamentaux restent identiques.

Comment procèdera-t-on alors ?

Décrivons d'abord l'information contenue dans le formulaire au moyen de l'objet DSL "ENTITY".

Pour la description des généralités sur le formulaire, on peut adopter une démarche analogue à celle rencontrée pour le message et pour le document:

- On peut définir globalement le formulaire au moyen de la clause DEFINE ENTITY.
  Exemple: DEFINE ENTITY demande-inscription.
- Chacun des attributs du formulaire sera décrit au moyen de la clause CONSISTS OF.
  Exemple:

```
DEFINE ENTITY  demande-inscription;
  CONSISTS OF  type-formulaire,
               identifiant-formulaire,
               état-formulaire,
               échéance-formulaire,
               statut-formulaire,
               support-formulaire;
```

- Pour la description proprement dite des attributs, on fera appel aux objets DSL "ELEMENT" et "GROUP".
  - On définira globalement les attributs au moyen des clauses DEFINE ELEMENT et DEFINE GROUP;
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.

  Exemple: pour l'attribut type-formulaire, on aura:

```
DEFINE ELEMENT type-formulaire;
  DOMAIN OF VALUE ARE  formulaire;
```

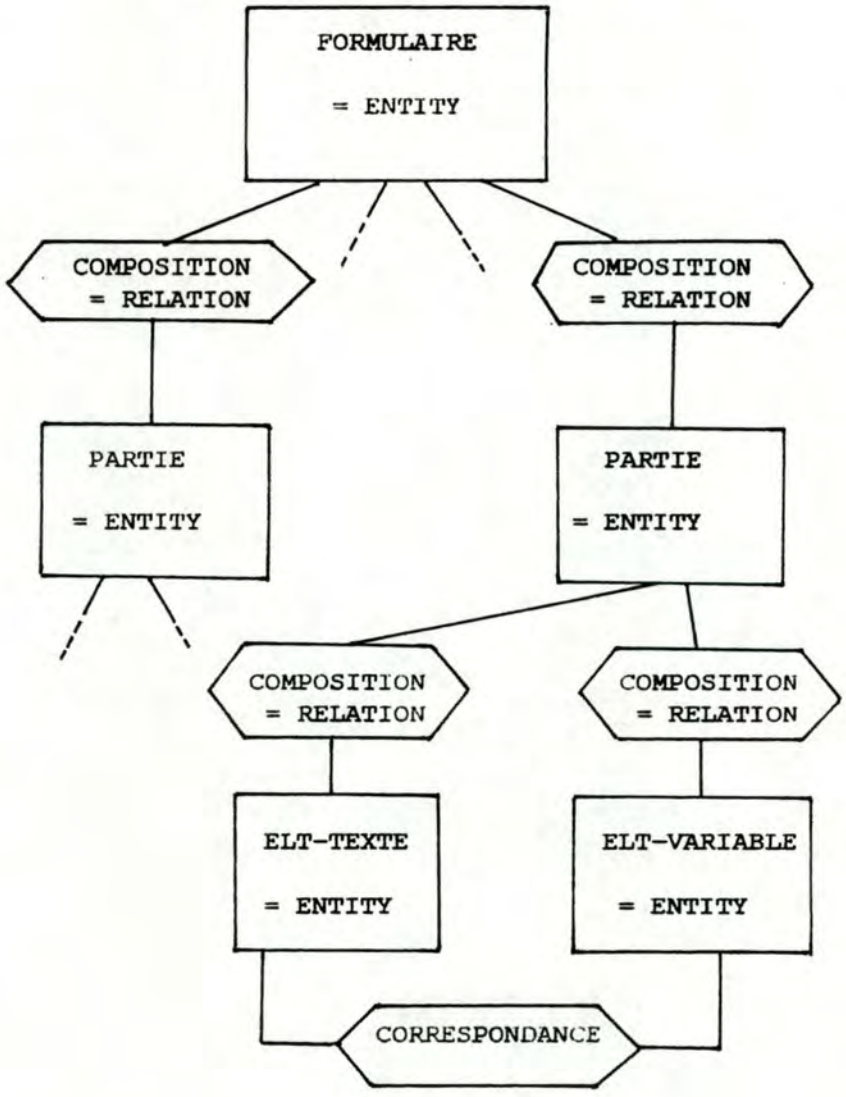
- Sachant que, dans les résultats, on voudrait pouvoir obtenir une liste de tous les formulaires décrits, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type de l'objet décrit.
  Exemple:

```
DEFINE ENTITY demande-inscription;
  KEYWORD ARE "FORMULAIRE";
```

En ce qui concerne le contenu du formulaire, on peut le représenter, comme dans le modèle proposé, par des relations avec la description générale du formulaire.

- Chaque partie, chaque élément du squelette textuel et chaque élément variable sera représenté au moyen de l'objet DSL "ENTITY".
  Les éléments de texte et les éléments variables ne peuvent en effet être représentés par les objets DSL "ELEMENT" et "GROUP" (ce qui semblerait pourtant plus logique) car ils possèdent eux-mêmes des attributs et sont reliés à la partie qui les contient par des associations ayant elles-mêmes des attributs. Les objets DSL "ELEMENT" et "GROUP" ne permettent pas de représenter ces contraintes.
- On définira la décomposition du formulaire en parties, la décomposition de chaque partie en éléments de texte et en éléments variables et la correspondance entre éléments de texte et éléments variables au moyen de l'objet DSL "RELATION".

Schématiquement, cela peut se représenter comme suit:



Après avoir décrit l'information contenue dans le formulaire au moyen de l'objet DSL "ENTITY", on le représente par autant d'objets DSL "MESSAGE" qu'il y a d'états dans son cycle de vie.

Doit-on représenter entièrement la description du formulaire à l'aide de l'objet DSL "MESSAGE", c'est-à-dire redécrire entièrement toute la décomposition du formulaire ?
Non. L'objet DSL "MESSAGE" permet de véhiculer l'information contenue dans le système d'information. Il servira, dans la dynamique, à déclencher des processus; il permet de connaitre l'origine et la destination de l'information. La description du formulaire à l'aide de l'objet DSL "MESSAGE" va donc nous permettre de décrire le chemin du formulaire dans le système d'information. A tout moment, ce qui a été décrit au moyen des objets DSL "ENTITY" sera disponible et ne devra donc plus apparaitre dans la description du formulaire à l'aide de l'objet DSL "MESSAGE". On réduira donc la description à l'aide de l'objet DSL "MESSAGE" à la desription générale du formulaire.

Pour chaque état:

- On définira globalement le message du MIB au moyen de la clause DEFINE MESSAGE.
  Exemple:
  ```
  DEFINE MESSAGE demande-inscription-rempli;
  DEFINE MESSAGE demande-inscription-classé-fich-inscr;
  ```
- Les attributs seront les mèmes que pour l'objet DSL "ENTITY". On les décrira au moyen de la clause CONSISTS OF sans les redécrire de façon individuelle.
- Dans la clause KEYWORD, on décrira encore le type de l'objet.
- Afin de faire le lien entre la représentation du formulaire à l'aide de l'objet DSL "MESSAGE" et sa représentation à l'aide de l'objet DSL "ENTITY", on utilisera la clause ATTRIBUTE IS pour définir l'attribut "est-associé-à".

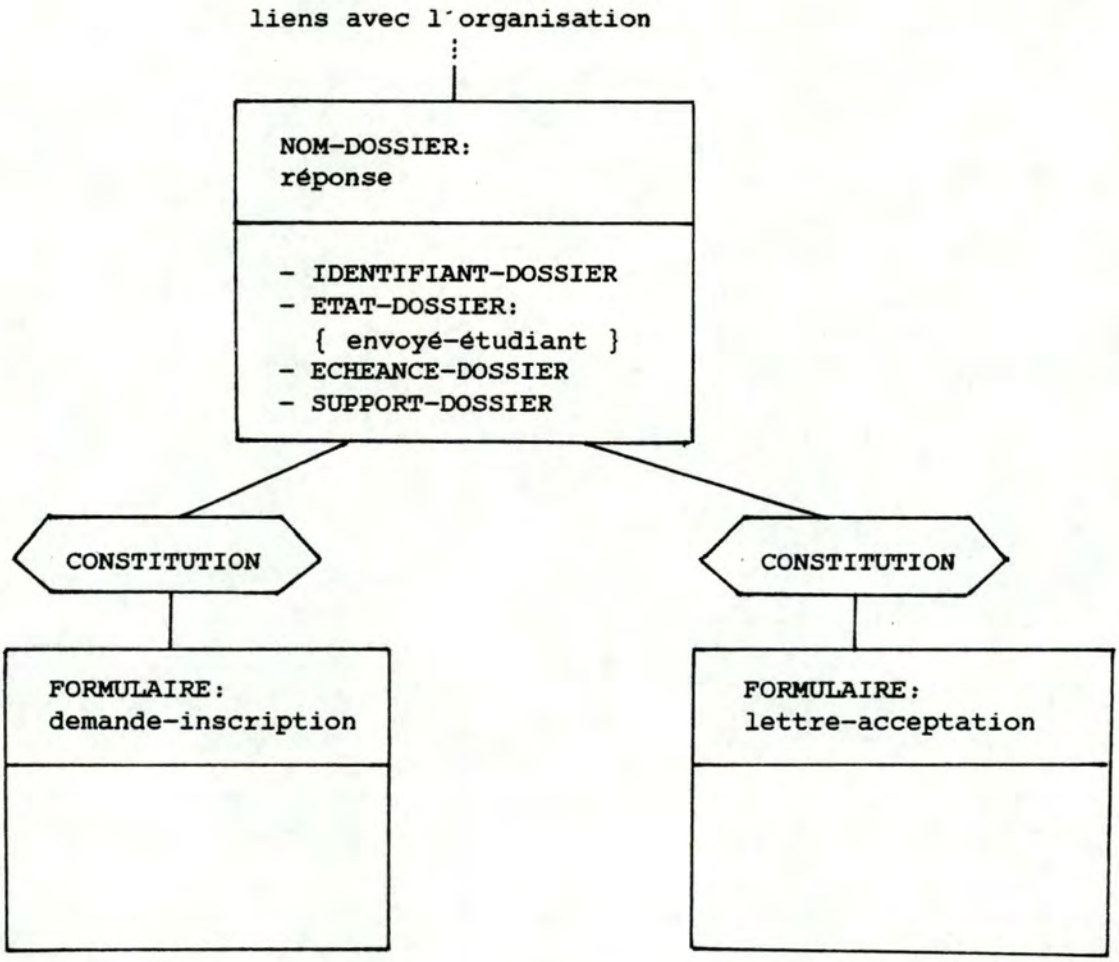
On peut trouver un exemple de transformation du formulaire en DSL à l'annexe 5 (Implémentation: Les rapports de la commande IP: A.3).

3 DESCRIPTION DU DOSSIER

Prenons à titre d'exemple le dossier réponse.

3.1 LE DOSSIER DANS LE MODELE PROPOSE

Dans le modèle proposé, on avait:



3.2 LE DOSSIER EN DSL

On va décrire d'abord l'information contenue dans le dossier au moyen de l'objet DSL "ENTITY":

- On peut définir globalement le dossier au moyen de la clause DEFINE ENTITY.
  Exemple: DEFINE ENTITY réponse.

- Chacun des attributs du dossier sera décrit au moyen de la clause CONSISTS OF.
  Exemple:
  ```
  DEFINE ENTITY  réponse;
    CONSISTS OF  identifiant-dossier,
                 état-dossier,
                 échéance-dossier,
                 support-dossier;
  ```
- Pour la description proprement dite des attributs, on fera appel aux objets DSL "ELEMENT" et "GROUP".
  - On définira globalement les attributs au moyen des clauses DEFINE ELEMENT et DEFINE GROUP.
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.
- Sachant que, dans les résultats, on voudrait pouvoir obtenir une liste de tous les dossiers décrits, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type de l'objet décrit.
  Exemple:
  ```
  DEFINE ENTITY réponse;
      KEYWORD ARE "DOSSIER";
  ```
- Connaissant les objets élémentaires constituant le dossier, on traduira cette contenance par des relations établies entre le dossier et les objets élémentaires qu'il contient.

Cette description du dossier au moyen de l'objet DSL "ENTITY" ne suffit pas. En effet, le dossier contient non seulement de l'information mais aussi véhicule cette information puisqu'il peut être communiqué.
On devra donc également décrire le dossier au moyen de l'objet DSL "MESSAGE", de façon analogue aux autres objets.
Pour chaque état:

- On définira globalement le dossier au moyen de la clause DEFINE MESSAGE.
  Exemple:
  ```
  DEFINE MESSAGE réponse-envoyé-étudiant;
  ```
- Les attributs seront les mêmes que pour l'objet DSL "ENTITY". On les décrira au moyen de la clause CONSISTS OF sans les redécrire de façon individuelle.
- Dans la clause KEYWORD, on décrira encore le type de l'objet.
- Afin de faire le lien entre la représentation du dossier à l'aide de l'objet DSL "MESSAGE" et sa représentation à l'aide de l'objet DSL "ENTITY", on utilisera la clause ATTRIBUTE IS pour définir l'attribut "est-associé-à".
- Les constituants du dossier ne doivent plus être décrits au moyen de l'objet DSL "MESSAGE".

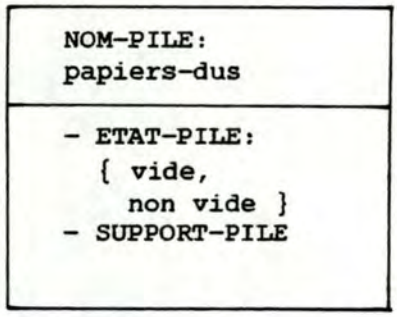
On peut trouver un exemple de transformation d'un dossier du modèle proposé en DSL à l'annexe 5 (Implémentation: Les rapports de la commandes IP: A.4).

4 DESCRIPTION DE LA PILE

Prenons comme exemple la pile papiers-dus.

4.1 LA PILE DANS LE MODELE PROPOSE

Dans le modèle proposé, on avait:



4.2 LA PILE EN DSL

On va décrire l'information contenue dans la pile au moyen de l'objet DSL "ENTITY":

- On peut définir globalement la pile au moyen de la clause DEFINE ENTITY.
  Exemple: DEFINE ENTITY papiers-dus.
- Chacun des attributs de la pile sera décrit au moyen de la clause CONSISTS OF.
  Exemple:

```
DEFINE ENTITY  papiers-dus;
  CONSISTS OF  état-pile
               support-pile;
```

- Pour la description proprement dite des attributs, on fera appel aux objets DSL "ELEMENT" et "GROUP".
  - On définira globalement les attributs au moyen des clauses DEFINE ELEMENT et DEFINE GROUP;
  - Les valeurs connues des attributs pourront être répertoriées dans la clause DOMAIN OF VALUE.
- Sachant que, dans les résultats, on voudrait pouvoir obtenir une liste de toutes les piles décrites, on liera tous les objets de ce type au moyen de la clause KEYWORD, dans laquelle on indiquera le type de l'objet décrit.
  Exemple:

```
DEFINE ENTITY papiers-dus;
    KEYWORD ARE "PILE"
```

On peut trouver un exemple de transformation d'une pile du modèle proposé en DSL à l'annexe 5 (Implémentation: Les rapports de la commandes IP: A.6).

ANNEXE 4: REPRESENTATION DES OBJETS INFORMATIONNELS EN DSL

Dans le chapitre 4, après avoir présenté une solution possible de description des objets du MIB à l'aide de DSL et après avoir donné la méthode de transformation de ces objets, on a décrit la représentation effective des objets en DSL. On s'était limité à la description de la représentation du document et du fichier. On va donner ici la représentation des autres objets.

1 REPRESENTATION EN DSL DU MESSAGE

1.1 DESCRIPTION DE L'ENREGISTREMENT

```
NOM-UTILISATEUR : ARRAY[1..12] OF CHAR;

MESSAGE : RECORD
  NOM : NOM-UTILISATEUR;
  TYPE : NOM-UTILISATEUR;
  EXPLICATION : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  EXPEDITEUR : NOM-UTILISATEUR;
  DESTINATAIRES : ARRAY[1..NBRE-DEST] OF NOM-UTILISATEUR;
  REPONSE : ARRAY[1..NBRE-REP] OF NOM-UTILISATEUR;
```

1.2 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Etant donné la solution choisie, on construira le texte DSL comme suit:

- On donnera le texte décrivant l'objet DSL "ENTITY" représentant le message du MIB.
- On donnera le texte décrivant les objets DSL "MESSAGE" représentant le message du MIB.
- On donnera le texte décrivant les liens avec les réponses attendues.
- On donnera le texte décrivant les liens avec l'organisation.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation du message du MIB à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans le message du MIB.

```
DEFINE ENTITY  MESSAGE.NOM_ENT;
  KEWORD ARE "MESGE";
  CONSISTS OF type-MESSAGE.NOM,
              explication-MESSAGE.NOM,
              contenu,
              état-MESSAGE.NOM,
              échéance,
              statut,
              support;
```

On décrit les attributs non prédéfinis au moyen de l'objet DSL "ELEMENT".

```
DEFINE ELEMENT  type-MESSAGE.NOM;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE MESSAGE.TYPE;

DEFINE ELEMENT  explication-MESSAGE.NOM;
  DOMAIN OF VALUE ARE MESSAGE.EXPLICATION;

DEFINE ELEMENT  état-MESSAGE.NOM;
  DOMAIN OF VALUE ARE  MESSAGE.ETAT[1],
        ...., MESSAGE.ETAT[NBRE-ETATS];
```

b. Représentation du message du MIB à l'aide de l'objet DSL "MESSAGE"

On décrit le message du MIB au moyen de l'objet DSL "MESSAGE" afin de permettre à l'information contenue dans le message du MIB de circuler dans le système d'infomation.
Nous avons vu que, si on veut simuler le comportement du système, on doit décrire autant d'objets DSL "MESSAGE" qu'il y a d'états dans le cycle de vie du message du MIB.

Pour chaque état du message du MIB, on décrira donc:

```
DEFINE MESSAGE  MESSAGE.NOM-MESSAGE.ETAT[i]_MES;
  KEYWORD ARE  "MESGE";
  ATTRIBUTE IS "associé-à"   "MESSAGE.NOM_ENT";
  CONSISTS OF  type-MESSAGE.NOM,
               contenu,
               échéance,
               statut,
               support;
```

c. Représentation des réponses attendues

On représente le lien entre l'objet communiqué et les objets attendus en réponse au moyen de l'objet DSL "RELATION".
Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans MESSAGE.NOM), du nom de l'objet attendu en réponse (contenu dans MESSAGE.REPONSE) et d'un mot désignant le lien: REP.

Pour chaque réponse attendue, on aura donc:

```
DEFINE RELATION  MESSAGE.NOM_REP_MESSAGE.REPONSE[i];
  RELATES  MESSAGE.NOM_ENT
    AS a-pour-réponse
    WITH CONNECTIVITY 0-N;
  RELATES  MESSAGE.REPONSE[i]_ENT
    AS répond-à
    WITH CONNECTIVITY 0-N;
```

d. Représentation des liens avec l'organisation

On connait le nom des expéditeurs et des destinataires.
On connait le nom de l'objet communiqué.
On définit une RELATION entre les objets DSL "ENTITY" représentant la description de l'objet communiqué et celle de l'expéditeur ou de chaque destinataire.

d.1. Représentation de l'expéditeur

Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans MESSAGE.NOM), du nom de l'expéditeur (contenu dans MESSAGE.EXPEDITEUR) et d'un mot désignant le lien: EXP.

```
DEFINE RELATION  MESSAGE.NOM_EXP_MESSAGE.EXPEDITEUR;
  CONSISTS OF date-exp-MESSAGE.NOM;
  RELATES MESSAGE.NOM_ENT
    AS expédié-par
    WITH CONNECTIVITY 1-1;
  RELATES MESSAGE.EXPEDITEUR_ENT
    AS expédie
    WITH CONNECTIVITY 0-N;

DEFINE ELEMENT  date-exp-MESSAGE.NOM;
  FORMAT IS 99/99/9999;
```

d.2. Représentation des destinataires

Le nom de la relation sera formé à partir du nom de l'objet communiqué, du nom du destinataire (contenu dans MESSAGE.DESTINATAIRES) et d'un mot désignant le lien: DEST.

Pour chaque destinataire, on aura:

```
DEFINE RELATION MESSAGE.NOM_DEST_MESSAGE.DESTINATAIRES[i];
  RELATES MESSAGE.NOM_ENT
    AS destiné-à
    WITH CONNECTIVITY 0-N;
  RELATES MESSAGE.DESTINATAIRES[i]_ENT
    AS reçoit
    WITH CONNECTIVITY 0-N;
```

2 REPRESENTATION EN DSL DU FORMULAIRE

2.1 DESCRIPTION DE L'ENREGISTREMENT

```
NOM-UTILISATEUR : ARRAY[1..12] OF CHAR;

FORMULAIRE : RECORD
  TITRE : NOM-UTILISATEUR;
  TYPE : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  EXPEDITEUR : NOM-UTILISATEUR;
  DESTINATAIRES : ARRAY[1..NBRE-DEST] OF NOM-UTILISATEUR;
  REPONSE : ARRAY[1..NBRE-REP] OF NOM-UTILISATEUR;
  DECOMPOSITION : ARRAY[1..NBRE-PART] OF PARTIE;

PARTIE : RECORD
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  EMPLACEMENT : ARRAY[1..2] OF INTEGER;
  SQUELETTE : ARRAY[1..NBRE-ELT-SQ] OF ELT-SQ-TXT;
  VARIABLE : ARRAY[1..NBRE-ELT-VAR] OF ELT-VAR;

ELT-SQ-TXT : RECORD
  NOM : NOM-UTILISATEUR;
  TEXTE : TEXTE;
  EMPLACEMENT : ARRAY[1..2] OF INTEGER;

ELT-VAR : RECORD
  NOM : NOM-UTILSATEUR;
  VAL-DEF : TEXTE;
  SQ-CORRESPONDANT : NOM-UTILISATEUR;
  EMPLACEMENT : ARRAY[1..2] OF INTEGER;
```

2.2 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Etant donné la solution choisie, on construira le texte DSL comme suit:

- On donnera le texte décrivant les objets DSL "ENTITY" représentant le formulaire et ses composants.
- On donnera le texte décrivant les objets DSL "MESSAGE" représentant le formulaire.
- On donnera le texte décrivant les liens avec les réponses attendues.
- On donnera le texte décrivant les liens avec l'organisation.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation du formulaire à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans le formulaire

a.1. Représentation des généralités concernant le formulaire

```
DEFINE ENTITY  FORMULAIRE.TITRE_ENT;
  KEWORD ARE "FORMULAIRE";
  CONSISTS OF type-FORMULAIRE.TITRE,
              identifiant-FORMULAIRE.TITRE,
              état-FORMULAIRE.TITRE,
              échéance,
              statut,
              support;

DEFINE ELEMENT type-FORMULAIRE.TITRE;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE FORMULAIRE.TYPE;

DEFINE ELEMENT identifiant-FORMULAIRE.TITRE;
  FORMAT IS alphabétique;

DEFINE ELEMENT état-FORMULAIRE.TITRE;
  DOMAIN OF VALUE ARE FORMULAIRE.ETAT[1],
       ...., FORMULAIRE.ETAT[NBRE-ETATS];
```

a.2.Représentation des composants du formulaire

Chaque partie sera représentée au moyen de l'objet DSL "ENTITY". On définira la décomposition du formulaire en parties au moyen de l'objet DSL RELATION".

Pour chaque partie, on aura:

```
DEFINE ENTITY FORMULAIRE.PARTIE[i].NOM_ENT;
  CONSISTS OF état-FORMULAIRE.PARTIE[i].NOM;

DEFINE ELEMENT état-FORMULAIRE.PARTIE[i].NOM;
  FORMAT IS alphabétique;
  DOMAIN OF VALUE ARE  FORMULAIRE.PARTIE[i].ETAT[1],
        ...., FORMULAIRE.PARTIE[i].ETAT[NBRE-ETATS];
```

```
DEFINE RELATION FORMULAIRE.TITRE_COMP_FORMULAIRE.PARTIE[i].NOM;
  CONSISTS OF emplacement-FORMULAIRE.PARTIE[i].NOM;
  RELATES FORMULAIRE.TITRE_ENT
    AS se-décompose-en
    WITH CONNECTIVITY 1-N;
  RELATES FORMULAIRE.PARTIE[i].NOM_ENT
    AS fait-partie-de
    WITH CONNECTIVITY 1-N;

DEFINE emplacement-FORMULAIRE.PARTIE[i].NOM;
  FORMAT IS 99.99;
  DOMAIN OF VALUE ARE
       FORMULAIRE.PARTIE[i].EMPLACEMENT[1]*
       FORMULAIRE.PARTIE[i].EMPLACEMENT[2];
```

On peut décrire de façon analogue la décomposition de chacune des parties du formulaire en éléments de texte et en éléments variables.

b. Représentation du formulaire à l'aide de l'objet DSL "MESSAGE"

On décrit le formulaire au moyen de l'objet DSL "MESSAGE" afin de permettre à l'information contenue dans le formulaire de circuler dans le système d'infomation.

Pour chaque état du formulaire, on aura:

```
DEFINE MESSAGE  FORMULAIRE.TITRE-FORMULAIRE.ETAT[i]_MES;
  KEYWORD ARE  "FORMULAIRE";
  ATTRIBUTE IS "associé-à"   "FORMULAIRE.TITRE_ENT";
  CONSISTS OF  type-FORMULAIRE.TITRE,
               identifiant-FORMULAIRE.TITRE,
               échéance,
               statut,
               support;
```

c. Représentation des réponses attendues

On représente le lien entre l'objet communiqué et les objets attendus en réponse au moyen de l'objet DSL "RELATION".
Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans FORMULAIRE.TITRE), du nom de l'objet attendu en réponse (contenu dans FORMULAIRE.REPONSE) et d'un mot désignant le lien: REP.

Pour chaque réponse attendue, on aura donc:

```
DEFINE RELATION  FORMULAIRE.TITRE_REP_FORMULAIRE.REPONSE[i];
  RELATES  FORMULAIRE.TITRE_ENT
    AS a-pour-réponse
    WITH CONNECTIVITY 0-N;
  RELATES  FORMULAIRE.REPONSE[i]_ENT
    AS répond-à
    WITH CONNECTIVITY 0-N;
```

d. Représentation des liens avec l'organisation

On connait le nom des expéditeurs et des destinataires.
On connait le nom de l'objet communiqué.
On définit une RELATION entre les objets DSL "ENTITY" représentant la description de l'objet communiqué et celle de l'expéditeur ou de chaque destinataire.

d.1. Représentation de l'expéditeur

Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans FORMULAIRE.TITRE), du nom de l'expéditeur (contenu dans FORMULAIRE.EXPEDITEUR) et d'un mot désignant le lien: EXP.

```
DEFINE RELATION  FORMULAIRE.TITRE_EXP_FORMULAIRE.EXPEDITEUR;
  CONSISTS OF date-exp-FORMULAIRE.TITRE;
  RELATES FORMULAIRE.TITRE_ENT
    AS expédié-par
    WITH CONNECTIVITY 1-1;
  RELATES FORMULAIRE.EXPEDITEUR_ENT
    AS expédie
    WITH CONNECTIVITY 0-N;

DEFINE ELEMENT  date-exp-FORMULAIR.TITRE;
  FORMAT IS 99/99/9999;
```

d.2. Représentation des destinataires

Le nom de la relation sera formé à partir du nom de l'objet communiqué, du nom du destinataire (contenu dans FORMULAIRE.DESTINATAIRES) et d'un mot désignant le lien: DEST.

Pour chaque destinataire, on aura:

```
DEFINE RELATION FORMULAIRE.TITRE_DEST_FORMULAIRE.DESTINATAIRES[i];
  RELATES FORMULAIRE.TITRE_ENT
    AS destiné-à
    WITH CONNECTIVITY 0-N;
  RELATES FORMULAIRE.DESTINATAIRES[i]_ENT
    AS reçoit
    WITH CONNECTIVITY 0-N;
```

3 REPRESENTATION EN DSL DU DOSSIER

3.1 DESCRIPTION DE L'ENREGISTREMENT

```
NOM-UTILISATEUR : ARRAY[1..12] OF CHAR;

DOSSIER : RECORD
  NOM : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
  EXPEDITEUR : NOM-UTILISATEUR;
  DESTINATAIRES : ARRAY[1..NBRE-DEST] OF NOM-UTILISATEUR;
  CONSTITUANTS : ARRAY[1..NBRE-CONS] OF NOM-UTILISATEUR;
```

3.2 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Etant donné la solution choisie, on construira le texte DSL comme suit:

- On donnera le texte décrivant l'objet DSL "ENTITY" représentant le dossier
- On donnera le texte décrivant les objets DSL "MESSAGE" représentant le dossier.
- On donnera le texte décrivant les liens avec les réponses attendues.
- On donnera le texte décrivant les liens avec l'organisation.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation du dossier à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans le dossier.

```
DEFINE ENTITY  DOSSIER.NOM_ENT;
  KEWORD ARE "DOSSIER";
  CONSISTS OF identifiant-DOSSIER.NOM,
              état-DOSSIER.NOM,
              échéance,
              support;
```

On décrit les attributs non prédéfinis au moyen de l'objet DSL "ELEMENT".

```
DEFINE ELEMENT  état-DOSSIER.NOM;
  DOMAIN OF VALUE ARE  DOSSIER.ETAT[1],
        ...., DOSSIER.ETAT[NBRE-ETATS];

DEFINE ELEMENT identifiant.DOSSIER.NOM;
  FORMAT IS alphabétique;
```

b. Représentation du dossier à l'aide de l'objet DSL "MESSAGE"

On décrit le dossier au moyen de l'objet DSL "MESSAGE" afin de permettre à l'information contenue dans le dossier de circuler dans le système d'infomation.

Pour chaque état du dossier, on décrira donc:

```
DEFINE MESSAGE  DOSSIER.NOM-DOSSIER.ETAT[i]_MES;
  KEYWORD ARE  "DOSSIER";
  ATTRIBUTE IS "associé-à"   "DOSSIER.NOM_ENT";
  CONSISTS OF  identifiant_DOSSIER.NOM,
               échéance,
               support;
```

c. Représentation des constituants du dossier

On connait le nom du dossier.
On connait le nom de ses constituants.
On définit une relation entre les objets DSL "ENTITY" représentant la description du dossier et celle de chaque constituant.

Pour chaque constituant, on aura:

```
DEFINE RELATION DOSSIER.NOM_CONS_DOSSIER.CONSTITUANT[i];
  RELATES DOSSIER.NOM_ENT
    AS contient
    WITH CONNECTIVITY 0-N;
  RELATES DOSSIER.CONSTITUANT[i]_ENT
    AS est-contenu-dans
    WITH CONNECTIVITY 0-1;
```

d. Représentation des liens avec l'organisation

On connait le nom des expéditeurs et des destinataires.
On connait le nom de l'objet communiqué.
On définit une RELATION entre les objets DSL "ENTITY" représentant la description de l'objet communiqué et celle de l'expéditeur ou de chaque destinataire.

d.1. Représentation de l'expéditeur

Le nom de la relation sera formé à partir du nom de l'objet communiqué (contenu dans DOSSIER.NOM), du nom de l'expéditeur (contenu dans DOSSIER.EXPEDITEUR) et d'un mot désignant le lien: EXP.

```
DEFINE RELATION  DOSSIER.NOM_EXP_DOSSIER.EXPEDITEUR;
  CONSISTS OF date-exp-DOSSIER.NOM;
  RELATES DOSSIER.NOM_ENT
    AS expédié-par
    WITH CONNECTIVITY 1-1;
  RELATES DOSSIER.EXPEDITEUR_ENT
    AS expédie
    WITH CONNECTIVITY 0-N;

DEFINE ELEMENT  date-exp-DOSSIER.NOM;
  FORMAT IS 99/99/9999;
```

d.2. Représentation des destinataires

Le nom de la relation sera formé à partir du nom de l'objet communiqué, du nom du destinataire (contenu dans DOSSIER.DESTINATAIRES) et d'un mot désignant le lien: DEST.

Pour chaque destinataire, on aura:

```
DEFINE RELATION DOSSIER.NOM_DEST_DOSSIER.DESTINATAIRES[i];
  RELATES DOSSIER.NOM_ENT
    AS destiné-à
    WITH CONNECTIVITY 0-N;
  RELATES DOSSIER.DESTINATAIRES[i]_ENT
    AS reçoit
    WITH CONNECTIVITY 0-N;
```

4 REPRESENTATION EN DSL DE LA PILE

4.1 DESCRIPTION DE L'ENREGISTREMENT

```
NOM-UTILISATEUR : ARRAY[1..12] OF CHAR;

PILE : RECORD
  NOM : NOM-UTILISATEUR;
  ETAT : ARRAY[1..NBRE-ETATS] OF NOM-UTILISATEUR;
```

4.2 DESCRIPTION DES SQUELETTES DE TEXTE DSL

Etant donné la solution choisie, on construira le texte DSL comme suit:

- On donnera le texte décrivant l'objet DSL "ENTITY" représentant la pile.

Dans ce texte DSL, on indiquera par quelle zone de l'enregistrement compléter les champs vides.

a. Représentation de la pile à l'aide de l'objet DSL "ENTITY"

L'objet DSL "ENTITY" nous permet de représenter l'information contenue dans la pile.

```
DEFINE ENTITY  PILE.NOM_ENT;
  KEWORD ARE "PILE";
  CONSISTS OF état-PILE.NOM,
              support;
```

On décrit les attributs non prédéfinis au moyen de l'objet DSL "ELEMENT".

```
DEFINE ELEMENT  état-PILE.NOM;
  DOMAIN OF VALUE ARE  PILE.ETAT[1],
        ...., PILE.ETAT[NBRE-ETATS];
```

ANNEXE 5: IMPLEMENTATION DU MIB.

1 PRESENTATION DES OUTILS UTILISES

Comme on l'a vu dans les résultats attendus du MIB, on voudrait aider l'utilisateur à spécifier ses objets de bureau. Cet objectif peut être atteint en lui offrant des écrans de description des objets de bureau. Pour ce, on utilisera le gestionnaire d'écran VAX-11 TDMS.

Les programmes de transformation des objets décrits, du MIB en DSL et inversément, seront écrits dans le langage C.

Le langage des commandes de DSL-SPEC permettra la création et l'exploitation de la base de données des spécifications en DSL.

1.1 LE GESTIONNAIRE D'ECRAN VAX-11 TDMS

L'utilitaire VAX-11 TDMS ([TDMS 1], [TDMS 2], [TDMS 3], [TDMS 4], [TDMS 5]) est un gestionnaire de données qui utilise des écrans ("forms") pour collecter et afficher de l'information au terminal.

TDMS remplace certaines parties d'un programme d'application par des définitions qui sont créées et rangées à l'extérieur de ce programme d'application. Ces définitions comprennent:

- des définitions d'écrans ("forms"):
  Elles définissent l'image qui apparait sur l'écran. Ces définitions incluent le contrôle des mouvements du curseur sur l'écran, la description de zones répétitives, des textes d'aide et d'explication et des caractéristiques vidéo. On peut aussi les utiliser pour restreindre les données que l'utilisateur peut introduire dans une zone de l'écran à l'exécution.

- des définitions d'enregistrements ("records"):
  Elles définissent les types de données, leur structure, leur ordre et la longueur des enregistrements que l'application utilise.
  Dans certains langages, et notamment en C, il n'est pas nécessaire de recopier ces enregistrements dans le programme d'application.

- des requêtes:
  Elles définissent les échanges d'information entre le programme (définitions d'enregistrement) et le terminal (définitions d'écrans), incluant l'input et l'output de données.

Les requêtes peuvent remplacer le codage des entrées / sorties qui devrait être requis dans le programme d'application.
Les requêtes identifient les écrans ("forms") à afficher au terminal et les données à collecter et / ou afficher sur ces écrans. On peut également utiliser les requêtes pour traiter certaines erreurs d'application.
Les requêtes sont groupées dans des fichiers de librairie des requêtes (RLB) qui incluent chaque requête et les définitions d'écrans et d'enregistrements nommées dans la requête.

A l'exécution, les appels de programmation TDMS exécutent les requêtes rangées dans les RLB. L'application est contrôlée par les requêtes que le programme exécute et les instructions spécifiées par ces requêtes. Les instructions sont principalement:

- l'affichage d'écran: DISPLAY;
- la collecte d'information: INPUT TO;
- l'affichage d'information: OUTPUT TO;
- l'exécution d'opérations conditionnelles.

1.2 LE LANGAGE C

Le langage C est un langage de programmation conçu pour de multiples applications.

Notre application se base surtout sur les entrées / sorties d'information. Les fonctions réalisant les entrées et les sorties n'appartiennent pas au langage C. Une "bibliothèque standard d'entrées / sorties" constitue un ensemble de fonctions qui fournit un système d'entrées / sorties standard pour les programmes écrits en langage C. Ces fonctions sont destinées à former un interface commode de programmation. Il existe notamment des fonctions d'entrées / sorties de caractères et de chaines de caractères, des fonctions d'entrées / sorties formatées, à partir du terminal ou à partir d'un fichier spécifié.

La bibliothèque offre aussi des fonctions de manipulation de chaines de caractères bien utiles: comparaison, concaténation, copie d'une chaine dans une autre, ...

La compatibilité du langage C avec l'utilitaire TDMS et les fonctions que ce langage offre nous ont amenés à le choisir comme langage de programmation de notre application.

1.3 LE LANGAGE DES COMMANDES DE DSL-SPEC

DSL-SPEC [IDA] désigne l'ensemble des programmes qui exploitent la base de données des spécifications en DSL.

DSL-SPEC stocke dans une base de données des objets, des relations et des informations descriptives qui lui sont fournies par l'utilisateur via des phrases DSL. Une fois ces informations dans la base, DSL-SPEC est capable de les modifier, d'en ajouter ou d'en supprimer, de même qu'il est capable de générer des rapports présentant l'état de la base de données, c'est-à-dire du problème décrit par l'utilisateur.

S'il connait le langage des commandes de DSL-SPEC, l'utilisateur peut manipuler aisément le contenu de la base de données.

Ces commandes du langage de commandes DSL-SPEC sont réparties en quatre grands groupes:

- les commandes de contrôle:
  On les utilise pour gérer l'environnement du système.

- les commandes de modification:
  On les utilise pour modifier le contenu de la base de données suivant les directives données par l'utilisateur.

- les commandes de production de rapport:
  On les utilise pour extraire des données de la base de données et les imprimer.

- le système d'interrogation de la base de données:
  Il a pour objectif d'extraire des ensembles d'objets de la base de données de spécification, qui vérifient des critères de sélection.

2 LE PROGRAMME "BUREAU"

2.1 INTRODUCTION

Le programme "BUREAU" permet la transformation de la description des objets de bureau du MIB en DSL. L'analyse complète de cette transformation a été réalisée au chapitre 4. Ce chapitre servira donc de point de départ à la phase de programmation que nous allons développer ici.

2.2 LES PRINCIPES DU PROGRAMME

Ce programme est une mise en oeuvre du schéma général de transformation des objets vu au chapitre 4 (point 3.1).

1. L'utilisateur décrit les objets du bureau selon le modèle de structuration des informations, via des écrans de saisie.
   Cette gestion d'écran est réalisée au moyen du gestionnaire de données VAX-11 TDMS.

2. L'information décrite par l'utilisateur est rangée dans des enregistrements.
   Ces enregistrements sont décrits et mis à jour également au moyen du gestionnaire TDMS (définition des enregistrements).

3. L'information décrite par l'utilisateur est partiellement vérifiée:
   _ des messages aidant l'utilisateur sont disponibles lors de la description des informations. En effet, TDMS permet d'afficher des messages et des écrans donnant de l'information sur la façon de remplir les différentes zones des écrans.

   _ TDMS permet également de donner des règles de validation de l'information décrite par l'utilisateur: exemple: les types du message et du document ne peuvent prendre que certaines valeurs définies.

   _ TDMS oblige l'utilisateur à remplir certains champs, permettant ainsi de réaliser la complétude du modèle.

4. L'information rangée dans les enregistrements est traitée de façon à transformer les objets de bureau décrits, en des objets DSL.
   Pour ce, on a défini l'information DSL à collecter: ce sont les "squelettes de texte DSL" dont on a parlé au chapitre 4. Ces squelettes de texte DSL seront complétés par l'information contenue dans les enregistrements.
   Cette transformation des objets du bureau en DSL est réalisée dans le langage C, sur VAX. Elle consiste principalement en des écritures, sur fichier, des objets en DSL.

5. Une fois que toute l'information fournie par l'utilisateur a été transfomée en DSL, on utilise les outils IDA pour ranger la description des objets en DSL dans la base de données DSL.

6. La description rangée dans la base de données DSL peut être exploitée. Comme on s'est seulement intéressé à la représentation des objets informationnels et non à la dynamique, la seule exploitation possible jusqu'à présent est la production de rapports documentaires.

2.3 LA DESCRIPTION DU PROGRAMME

2.3.1 LA GESTION D'ECRAN AU MOYEN DE TDMS

La première étape dans l'élaboration du programme a été de définir les informations à décrire. Ces informations constitueront les données principales du programme.

La gestion d'écran a été réalisée en utilisant le gestionnaire d'écran TDMS. Comme on l'a vu dans la présentation des outils, cette gestion comporte trois phases:

- A. la définition des écrans;
- B. la définitions des enregistrements;
- C. les requêtes.

A. LA DEFINITION DES ECRANS

Il s'agit ici de décrire les écrans tels qu'ils apparaitront à l'utilisateur. Pour ce, il faut déterminer les données à saisir, leur type et les restrictions éventuelles.
On va maintenant décrire les écrans utilisés par le programme "BUREAU".

a. l'écran "INITIAL_BUREAU_FORM"

Cet écran donne à l'utilisateur une description générale du programme "BUREAU"
Il se présente comme suit:

CECI EST LE PROGRAMME

BUREAU

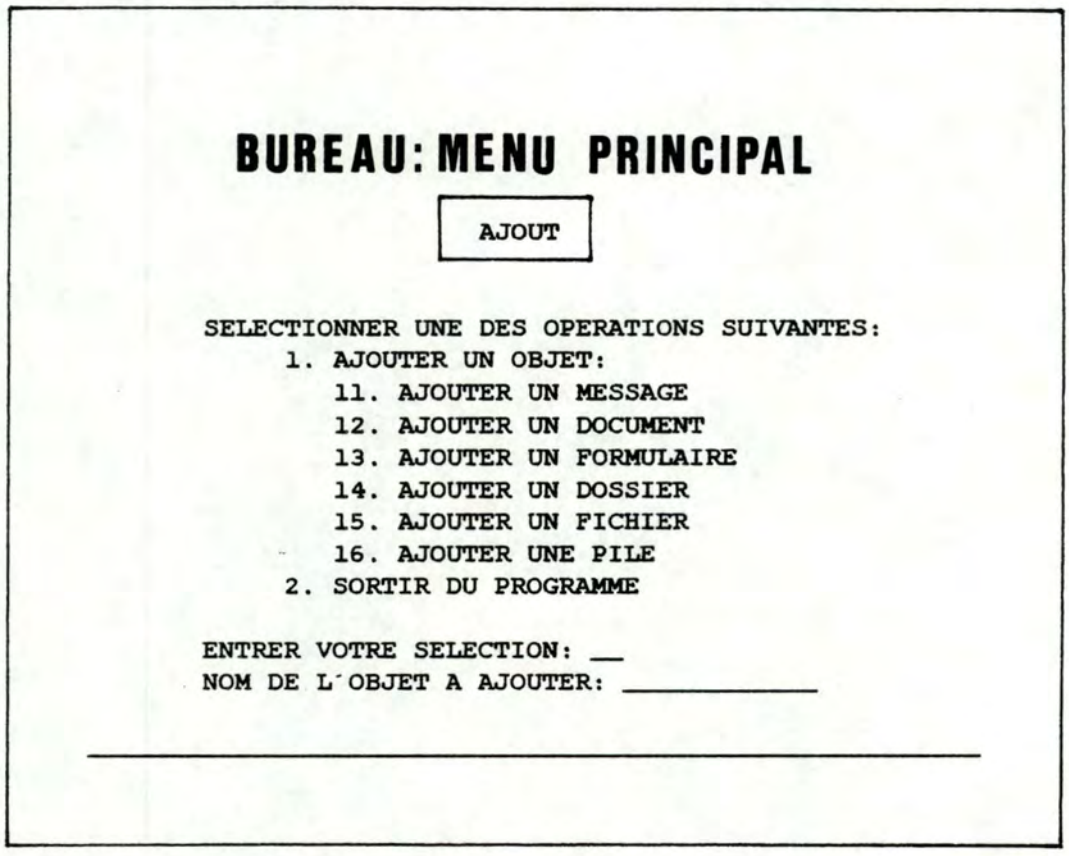
Ce programme vous permet de décrire les objets de bureau, c'est-à-dire :
_ les objets élémentaires: message, document, formulaire;
_ les objets structurants: dossier, fichier, pile;
ainsi que les relations:
_ entre ces objets: réponses attendues, classement dans les objets structurants;
_ entre ces objets et l'organisation: expéditeur, destinataires.

Dans le menu principal, il vous suffira de donner le numéro de l'objet que vous désirez décrire.

Pour chaque écran où vous devez donner de l'information, vous pouvez faire appel à une aide en appuyant sur la touche PF2 du clavier auxiliaire.

b. l'écran "MENU_BUREAU_FORM"

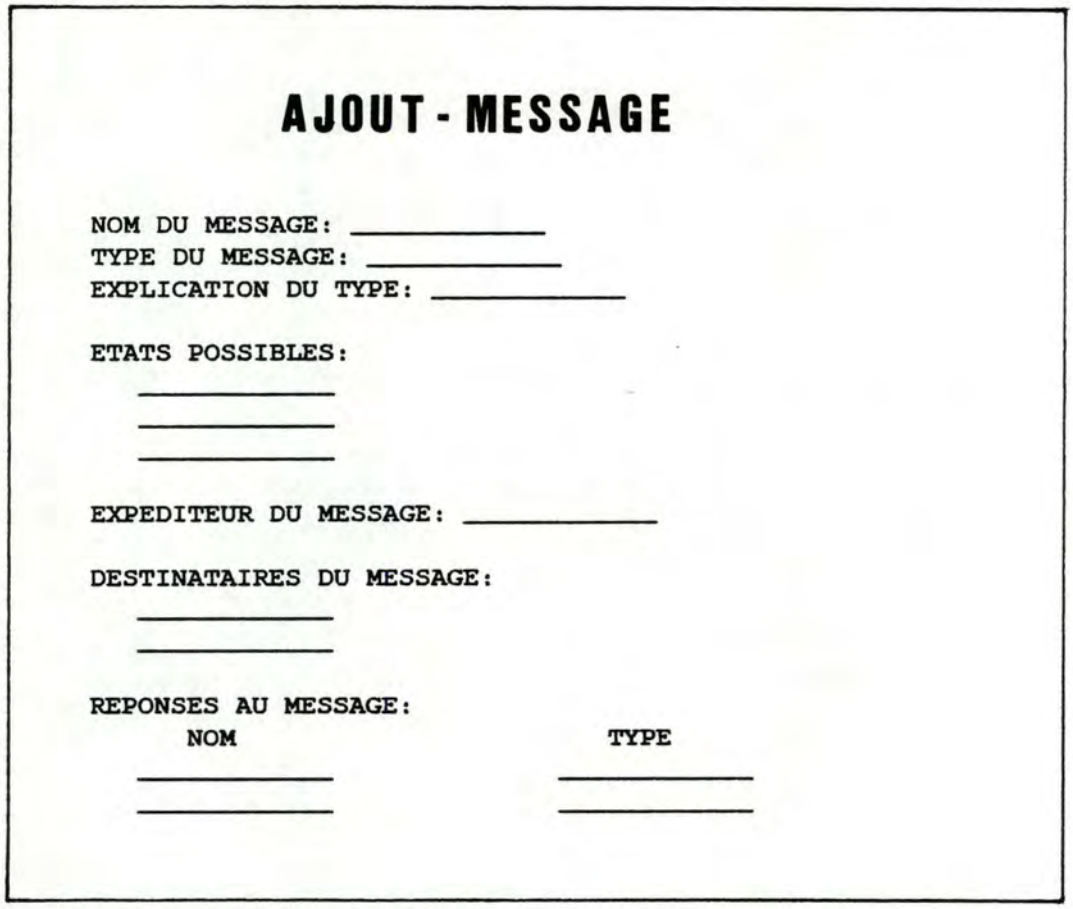
Cet écran permet d'afficher le menu du programme. Il se présente comme suit:



L'utilisateur doit donner le chiffre (donnée numérique) correspondant à son choix et le nom de l'objet (donnée alphanumérique) qu'il désire décrire. Une zone est également réservée en bas d'écran pour afficher des messages d'erreur, suite au choix de l'utilisateur.

c. l'écran "AJOUT_MESSAGE_FORM"

Cet écran permet à l'utilisateur de décrire un message. Il se présente comme suit:



Description des différentes zones à remplir:

_ NOM DU MESSAGE:
  * nom TDMS de la zone: NOM_MESS.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ TYPE DU MESSAGE:
  * nom TDMS de la zone: TYPE_MESS.
  * 12 caractères alphanumériques.
  * L'utilisateur peut donner uniquement les valeurs: téléphone, memo, réunion, telex, télégramme, avis, notice, calendrier, autre.
  * Cette zone doit obligatoirement être remplie.

_ EXPLICATION DU TYPE:
  * nom TDMS de la zone: EXPLICATION_MESS.
  * 12 caractères alphanumériques.
  * L'utilisateur peut y désigner un type non standard correspondant au type "autre".

_ ETATS DU MESSAGE:
  * nom TDMS de la zone: ETATS_MESS.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois; Trois répétitions apparaissent simultanément à l'écran.

_ EXPEDITEUR DU MESSAGE:
  * nom TDMS de la zone: EXPED_MESS.
  * 12 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ DESTINATAIRES DU MESSAGE:
  * nom TDMS de la zone: DEST_MESS.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois; Deux répétitions apparaissent simultanément à l'écran.

_ REPONSES AU MESSAGE:
_ NOM:
  * nom TDMS de la zone: NOM.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois; Deux répétitions apparaissent simultanément à l'écran.
_ TYPE:
  * nom TDMS de la zone: TYPE.
  * 10 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois; Deux répétitions apparaissent simultanément à l'écran.
  * Les valeurs possibles sont: message, document ou formulaire.

d. l'écran "HELPMESS"

Cet écran est affiché si l'utilisateur demande des informations sur la façon de remplir l'écran "AJOUT_MESSAGE_FORM".
Il se présente comme suit:

HELP: AJOUT-MESSAGE

Le nom du message doit se composer de 12 caractères minuscules.

Le type du message peut prendre les valeurs suivantes:
téléphone, memo, réunion, telex, télégramme, avis, notice, calendrier, autre.
Si le type est "autre", on pourra donner une description de ce type.

Les états possibles comprendront 12 caractères minuscules au maximum.
Le programme permet de spécifier au maximum 10 états.

Le nom de l'expéditeur est une valeur à fournir obligatoirement.

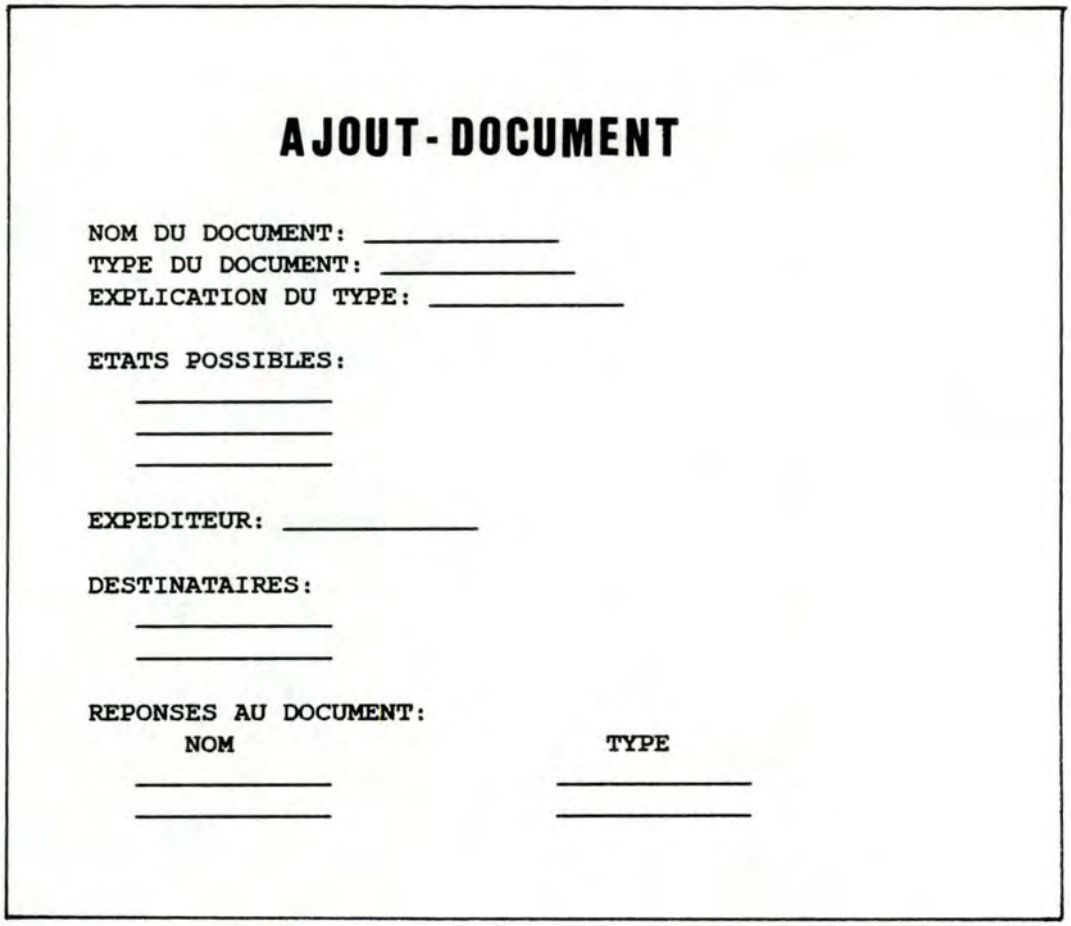
Le programme permet de définir au maximum 10 destinataires.

On donnera les réponses attendues suite à la communication d'un objet, et le type de ces réponses.
On pourra décrire au maximum 10 réponses.

Cet écran est descriptif et ne comporte aucune zone à compléter.

e. l'écran "AJOUT_DOCUMENT_FORM"

Cet écran permet à l'utilisateur de décrire un document. Il se présente comme suit:



Description des différentes zones à remplir:

_ NOM DU DOCUMENT:
  * nom TDMS de la zone: NOM_DOC.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ TYPE DU DOCUMENT:
  * nom TDMS de la zone: TYPE_DOC.
  * 12 caractères alphanumériques.
  * L'utilisateur peut donner uniquement les valeurs: lettre, brochure, journal, article, rapport, livre, projet, autre.
  * Cette zone doit obligatoirement être remplie.

_ EXPLICATION DU DOCUMENT:
  * nom TDMS de la zone: EXPLICATION_DOC.
  * 12 caractères alphanumériques.
  * L'utilisateur peut y désigner un type non standard correspondant au type "autre".

_ ETATS DU DOCUMENT:
  * nom TDMS de la zone: ETATS_DOC.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Trois répétitions apparaissent simultanément à l'écran.

_ EXPEDITEUR:
  * nom TDMS de la zone: EXPED_DOC.
  * 12 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ DESTINATAIRES:
  * nom TDMS de la zone: DEST_DOC.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.

_ REPONSES AU DOCUMENT:
_ NOM:
  * nom TDMS de la zone: NOM.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.
_ TYPE:
  * nom TDMS de la zone: TYPE.
  * 10 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.
  * Les valeurs possibles sont: message, document ou formulaire.

f. l'écran "HELPDOC"

Cet écran est affiché si l'utilisateur demande des informations sur la façon de remplir l'écran "AJOUT_DOCUMENT_FORM".
Il se présente comme suit:

HELP: AJOUT-DOCUMENT

Le nom du document doit se composer de 12 caractères minuscules.

Le type du document peut prendre les valeurs suivantes:
lettre, brochure, journal, article, rapport, livre, projet, autre.
Si le type est "autre", on pourra donner une description de ce type.

Les états possibles comprendront 12 caractères minuscules au maximum.
Le programme permet de spécifier au maximum 10 états.

Le nom de l'expéditeur est une valeur à fournir obligatoirement.

Le programme permet de définir au maximum 10 destinataires.

On donnera les réponses attendues suite à la communication d'un objet, et le type de ces réponses.
On pourra décrire au maximum 10 réponses.

Cet écran est descriptif et ne comporte aucune zone à compléter.

g. l'écran "FORMULAIRE_EXPLICATION_FORM"

Cet écran donne des informations sur la façon dont se déroule la description complète d'un formulaire.

AJOUT-FORMULAIRE.
EXPLICATION.

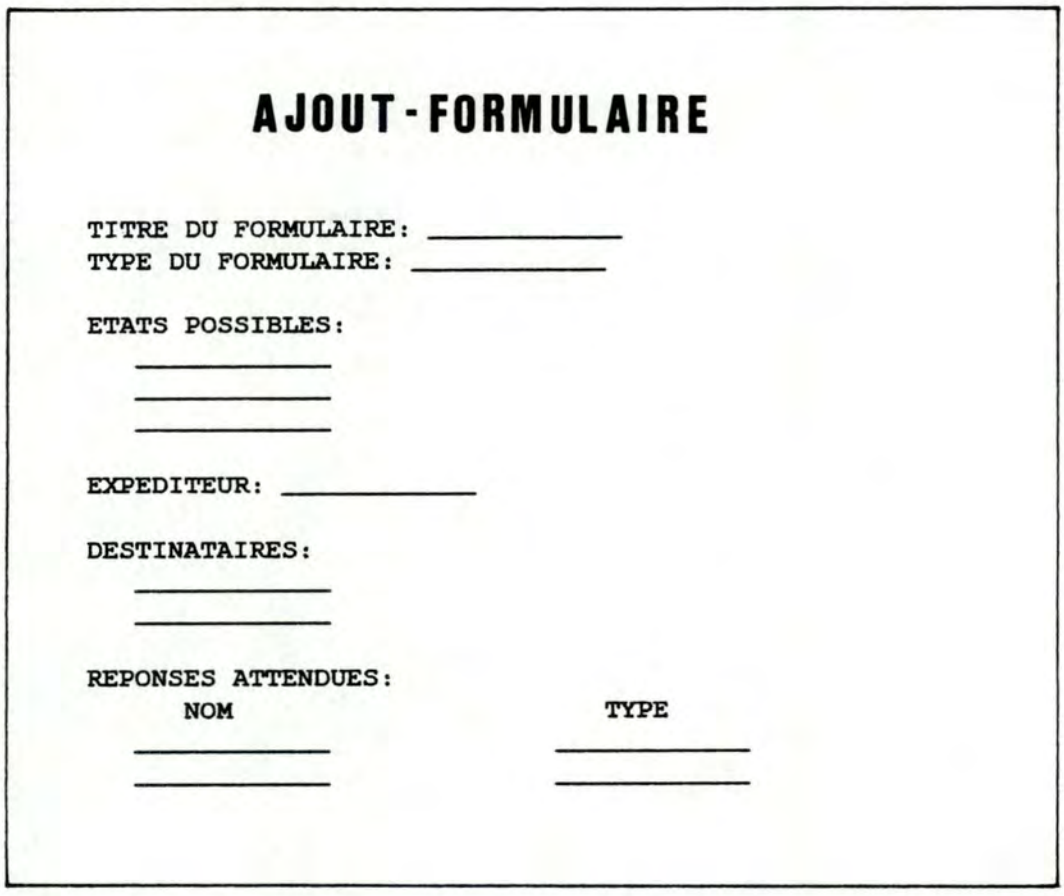
Pour ajouter le formulaire, on procèdera comme suit:

1. Entrer les informations générales sur le formulaire.

2. Détailler le formulaire, PARTIE par PARTIE:
   Pour chaque partie, on donnera:
   2.1. Les renseignements généraux sur la partie;
   2.2. Les éléments de texte qui la composent;
   2.3. Les éléments variables qui la composent.

h. l'écran "AJOUT_FORMULAIRE_FORM"

Cet écran permet à l'utilisateur de décrire un formulaire. Il se présente comme suit:



Description des différentes zones à remplir:

_ TITRE DU FORMULAIRE:
  * nom TDMS de la zone: TITRE_FORM.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ TYPE DU FORMULAIRE:
  * nom TDMS de la zone: TYPE_FORM.
  * 10 caractères alphanumériques.
  * L'utilisateur peut donner uniquement les valeurs: formulaire et liste.
  * Cette zone doit obligatoirement être remplie.

_ ETATS DU FORMULAIRE:
  * nom TDMS de la zone: ETATS_FORM.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Trois répétitions apparaissent simultanément à l'écran.

_ EXPEDITEUR:
  * nom TDMS de la zone: EXPED_MESS.
  * 12 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ DESTINATAIRES:
  * nom TDMS de la zone: DEST_FORM.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.

_ REPONSES ATTENDUES:
_ NOM:
  * nom TDMS de la zone: NOM.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.
_ TYPE:
  * nom TDMS de la zone: TYPE.
  * 10 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.
  * Les valeurs possibles sont: message, document ou formulaire.

i. l'écran "AJOUT_PARTIE_FORM"

Cet écran permet à l'utilisateur de décrire une partie d'un formulaire.
Il se présente comme suit:

```
                AJOUT-FORMULAIRE: PARTIE.


    NOM DE LA PARTIE: ____________

    EMPLACEMENT DE LA PARTIE:
       LIGNE.              COLONNE.
         99                   99
         99                   99
         99                   99

    ETATS POSSIBLES:
       ____________
       ____________
       ____________

    TAPEZ  T  SI VOUS DESIREZ ENTRER UN ELEMENT DE TEXTE.
    TAPEZ  V  SI VOUS DESIREZ ENTRER UN ELEMENT VARIABLE.

    VOTRE CHOIX: _
```

Description des différentes zones à remplir:

_ NOM DE LA PARTIE:
  * nom TDMS de la zone: NOM_PARTIE.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ EMPLACEMENT DE LA PARTIE:
_ LIGNE:
  * nom TDMS de la zone: LIGNE_PART.
  * numérique non signé.
  * Cette zone peut etré répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.
_ COLONNE:
  * nom TDMS de la zone: COLONNE_PART.
  * numérique non signé.
  * Cette zone peut être répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.

_ ETATS DE LA PARTIE:
  * nom TDMS de la zone: ETATS_PARTIE.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Trois répétitions apparaissent simultanément à l'écran.

_ CHOIX:
  * nom TDMS de la zone: SELECT_ELEMENT.
  * 1 caractère alphabétique.
  * Les valeurs possibles de cette zone sont: T ou V.

j. l'écran "AJOUT_TEXTE_FORM"

Cet écran permet à l'utilisateur de décrire un élément du squelette textuel d'un formulaire.
Il se présente comme suit:

AJOUT-FORMULAIRE
SQUELETTE TEXTUEL

NOM DE L'ELEMENT DU SQUELETTE TEXTUEL: _____
FORMAT: _____
TEXTE: _____

EMPLACEMENT DE L'ELEMENT TEXTUEL:

LIGNE.	COLONNE.
99	99
99	99
99	99

TAPEZ T SI VOUS DESIREZ ENTRER UN ELEMENT DE TEXTE.
TAPEZ V SI VOUS DESIREZ ENTRER UN ELEMENT VARIABLE.
TAPEZ P SI VOUS DESIREZ PASSER A UNE AUTRE PARTIE.
TAPEZ E (EXIT) SI VOUS AVEZ TERMINE.

VOTRE CHOIX: _

Description des différentes zones à remplir:

_ NOM DE L'ELEMENT DU SQUELETTTE TEXTUEL:
  * nom TDMS de la zone: NOM_TEXTE.
  * 12 caractères alphanumériques.

_ FORMAT:
  * nom TDMS de la zone: FORMAT.
  * 30 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ TEXTE:
  * nom TDMS de la zone: CONTENU.
  * 30 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ EMPLACEMENT DE L'ELEMENT TEXTUEL:
_ LIGNE:
  * nom TDMS de la zone: LIGNE_TXT.
  * numérique non signé.
  * Cette zone peut etré répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.
_ COLONNE:
  * nom TDMS de la zone: COLONNE_TXT.
  * numérique non signé.
  * Cette zone peut être répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.

_ CHOIX:
  * nom TDMS de la zone: SELECT_ELEMENT.
  * 1 caractère alphabétique.
  * Les valeurs possibles de cette zone sont: T, V, P ou E.

k. l'écran "AJOUT_VARIABLE_FORM"

Cet écran permet à l'utilisateur de décrire un élément variable d'un formulaire.
Il se présente comme suit:

```
                AJOUT-FORMULAIRE
                ELEMENT VARIABLE


    NOM DE L'ELEMENT VARIABLE: _____________
    FORMAT: ____________________________
    VALEUR PAR DEFAUT: ____________________________

    EMPLACEMENT DE L'ELEMENT VARIABLE:
       LIGNE.            COLONNE.
         99                 99
         99                 99
         99                 99

    ELEMENT TEXTUEL CORRESPONDANT: _____________

    TAPEZ  T  SI VOUS DESIREZ ENTRER UN ELEMENT DE TEXTE.
    TAPEZ  V  SI VOUS DESIREZ ENTRER UN ELEMENT VARIABLE.
    TAPEZ  P  SI VOUS DESIREZ PASSER A UNE AUTRE PARTIE.
    TAPEZ  E  ( EXIT )  SI VOUS AVEZ TERMINE.

    VOTRE CHOIX: _
```

Description des différentes zones à remplir:

_ NOM DE L'ELEMENT VARIABLE:
  * nom TDMS de la zone: NOM_VAR.
  * 12 caractères alphanumériques.

_ FORMAT:
  * nom TDMS de la zone: FORMAT.
  * 30 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ VALEUR PAR DEFAUT:
  * nom TDMS de la zone: VAL_DEF.
  * 30 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

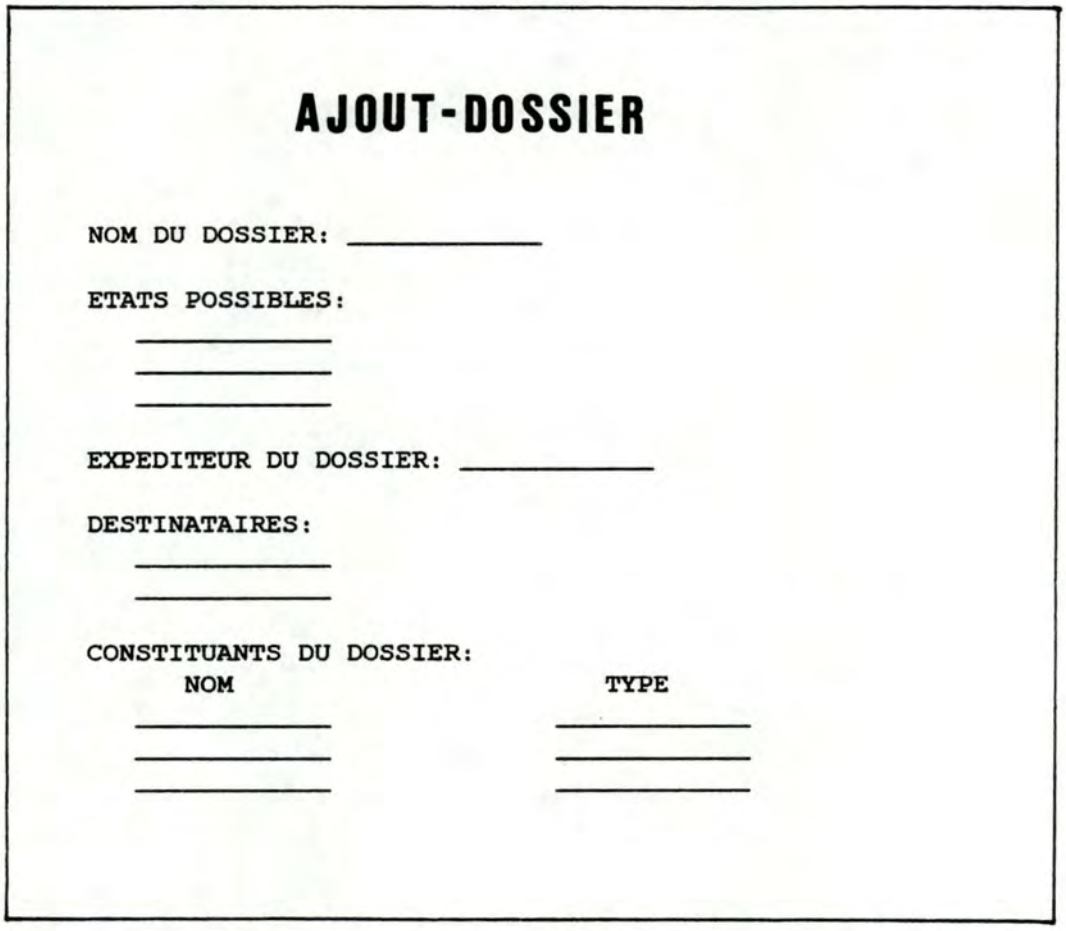
_ EMPLACEMENT DE L'ELEMENT VARIABLE:
_ LIGNE:
  * nom TDMS de la zone: LIGNE_VAR.
  * numérique non signé.
  * Cette zone peut etré répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.
_ COLONNE:
  * nom TDMS de la zone: COLONNE_VAR.
  * numérique non signé.
  * Cette zone peut être répétée dix fois.
    Trois répétitions apparaissent simultanément à l'écran.

_ ELEMENT TEXTUEL CORRESPONDANT:
  * nom TDMS de la zone: TXT_COR.
  * 12 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

_ CHOIX:
  * nom TDMS de la zone: SELECT_ELEMENT.
  * 1 caractère alphabétique.
  * Les valeurs possibles de cette zone sont: T, V, P ou E.

1. l'écran "AJOUT_DOSSIER_FORM"

Cet écran permet à l'utilisateur de décrire un dossier. Il se présente comme suit:



Description des différentes zones à remplir:

_ NOM DU DOSSIER:
  * nom TDMS de la zone: NOM_DOSSIER.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ ETATS DU DOSSIER:
  * nom TDMS de la zone: ETATS_DOSS.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois.
    Trois répétitions apparaissent simultanément à l'écran.

_ EXPEDITEUR:
  * nom TDMS de la zone: EXPED_DOSS.
  * 12 caractères alphanumériques.

_ DESTINATAIRES:
  * nom TDMS de la zone: DEST_DOSS.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée dix fois.
    Deux répétitions apparaissent simultanément à l'écran.

_ CONSTITUANTS DU DOSSIER:
_ NOM:
  * nom TDMS de la zone: CONST_DOSS.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée dix fois.
    Deux répétitions apparaissent simultanément à l'écran.
_ TYPE:
  * nom TDMS de la zone: TYPE_CONST_DOSS.
  * 10 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois;
    Deux répétitions apparaissent simultanément à l'écran.
  * Les valeurs possibles sont: message, document ou formulaire.

m. l'écran "AJOUT_FICHIER_FORM"

Cet écran permet à l'utilisateur de décrire un fichier. Il se présente comme suit:

Description des différentes zones à remplir:

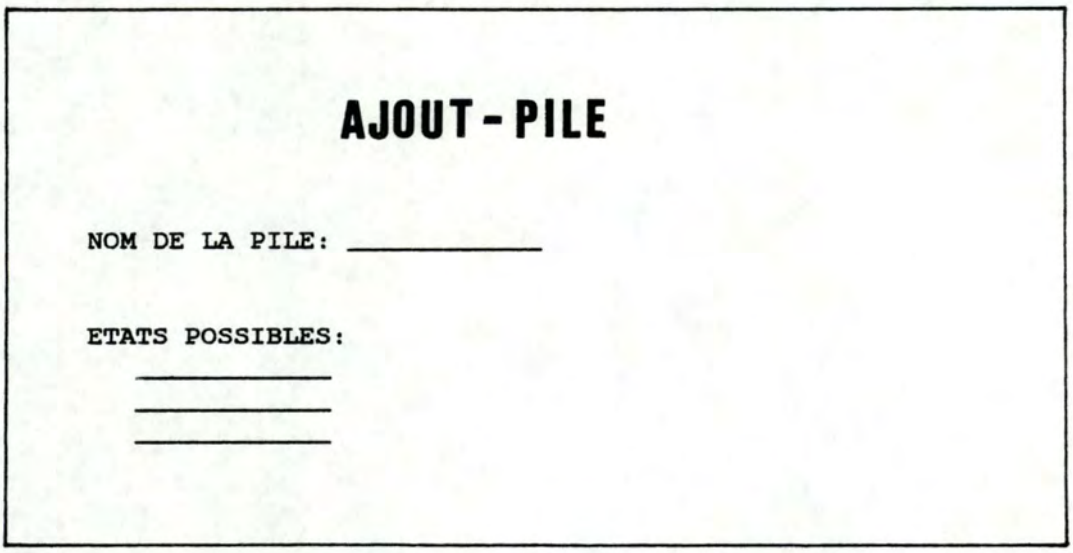
_ NOM DU FICHIER:
  * nom TDMS de la zone: NOM_FICHIER.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ ETATS DU FICHIER:
  * nom TDMS de la zone: ETATS_FICH.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à dix fois.
    Trois répétitions apparaissent simultanément à l'écran.

_ CONSTITUANT DU FICHIER:
_ NOM:
  * nom TDMS de la zone: CONST_FICH.
  * 12 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.
_ TYPE:
  * nom TDMS de la zone: TYPE_CONST_FICH.
  * 10 caractères alphanumériques.
  * Cette zone doit obligatoirement être remplie.

n. l'écran "AJOUT_PILE_FORM"

Cet écran permet à l'utilisateur de décrire une pile. Il se présente comme suit:



Description des différentes zones à remplir:

_ NOM DE LA PILE:
  * nom TDMS de la zone: NOM_PILE.
  * 12 caractères alphanumériques.
  * Cette zone est affichée par le programme.

_ ETATS DE LA PILE:
  * nom TDMS de la zone: ETATS_PILE.
  * 12 caractères alphanumériques.
  * Cette zone peut être répétée jusqu'à 10 fois.
    Trois répétitions apparaissent simultanément à l'écran.

B. LA DEFINITION DES ENREGISTREMENTS DE TDMS

On définit les types, la structure, l'ordre et la longueur des données. L'information décrite au moyen des écrans viendra se ranger dans ces enregistrements.

a. l'enregistrement "BUREAU_TRAVAIL_RECORD"

Cet enregistrement contient différentes variables de travail utilisées dans l'application, notamment les variables décrites dans le menu de l'application et des variables permettant la gestion d'erreur.

Description de l'enregistrement:

```
DEFINE RECORD BUREAU_TRAVAIL_RECORD.
   BUREAU_TRAVAIL_REC STRUCTURE.
      SELECTION                    DATATYPE SIGNED LONGWORD.
      MESS_ERREUR                  DATATYPE TEXT        80.
      CONSULT_OBJET                DATATYPE TEXT        12.
      PROGRAM_REQUEST_KEY          DATATYPE TEXT        12.
      VERIF_DONNEES                DATATYPE TEXT         1.
      SELECT_ELEMENT               DATATYPE TEXT         1.
   END BUREAU_TRAVAIL_REC STRUCTURE.
END BUREAU_TRAVAIL_RECORD.
```

b. l'enregistrement "MESSAGE_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "message".

Description de l'enregistrement:

```
DEFINE RECORD MESSAGE_RECORD.
   MESSAGE_REC STRUCTURE.
      NOM_MESS                  DATATYPE TEXT      12.
      TYPE_MESS                 DATATYPE TEXT      12.
      EXPLICATION_MESS          DATATYPE TEXT      12.
      M_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_MESS             DATATYPE TEXT      12.
         DUM_ETAT               DATATYPE TEXT       1.
      END M_ETATS_POSS STRUCTURE.
      EXPED_MESS                DATATYPE TEXT      12.
      M_DESTINATAIRES STRUCTURE OCCURS 10 TIMES.
         DEST_MESS              DATATYPE TEXT      12.
         DUM_DEST               DATATYPE TEXT       1.
      END M_DESTINATAIRES STRUCTURE.
      REPONSE_MESS STRUCTURE OCCURS 10 TIMES.
         NOM                    DATATYPE TEXT      12.
         TYPE                   DATATYPE TEXT      10.
         DUM_REP                DATATYPE TEXT       1.
      END REPONSE_MESS STRUCTURE.
   END MESSAGE_REC STRUCTURE.
END MESSAGE_RECORD.
```

c. l'enregistrement "DOCUMENT_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "document".
Description de l'enregistrement:

```
DEFINE RECORD DOCUMENT_RECORD.
   DOCUMENT_REC STRUCTURE.
      NOM_DOC                    DATATYPE TEXT      12.
      TYPE_DOC                   DATATYPE TEXT      12.
      EXPLICATION_DOC            DATATYPE TEXT      12.
      D_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_DOC               DATATYPE TEXT      12.
         DUM_ETAT                DATATYPE TEXT       1.
      END D_ETATS_POSS STRUCTURE.
      EXPED_DOC                  DATATYPE TEXT      12.
      D_DESTINATAIRES STRUCTURE OCCURS 10 TIMES.
         DEST_DOC                DATATYPE TEXT      12.
         DUM_DEST                DATATYPE TEXT       1.
      END D_DESTINATAIRES STRUCTURE.
      REPONSE_DOC STRUCTURE OCCURS 10 TIMES.
         NOM                     DATATYPE TEXT      12.
         TYPE                    DATATYPE TEXT      10.
         DUM_REP                 DATATYPE TEXT       1.
      END REPONSE_DOC STRUCTURE.
   END DOCUMENT_REC STRUCTURE.
END DOCUMENT_RECORD.
```

d. l'enregistrement "FORMULAIRE_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "formulaire".
Description de l'enregistrement:

```
DEFINE RECORD FORMULAIRE_RECORD.
   FORMULAIRE_REC STRUCTURE.
      TITRE_FORM                 DATATYPE TEXT      12.
      TYPE_FORM                  DATATYPE TEXT      10.
      F_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_FORM              DATATYPE TEXT      12.
         DUM_ET_FORM             DATATYPE TEXT       1.
      END F_ETATS_POSS STRUCTURE.
      EXPED_FORM                 DATATYPE TEXT      12.
      F_DESTINATAIRES STRUCTURE OCCURS 10 TIMES.
         DEST_FORM               DATATYPE TEXT      12.
         DUM_DES_FORM            DATATYPE TEXT       1.
      END F_DESTINATAIRES STRUCTURE.
      REPONSE_FORM STRUCTURE OCCURS 10 TIMES.
         NOM                     DATATYPE TEXT      12.
         TYPE                    DATATYPE TEXT      10.
         DUM_REP                 DATATYPE TEXT       1.
      END REPONSE_FORM STRUCTURE.
   END FORMULAIRE_REC STRUCTURE.
END FORMULAIRE_RECORD.
```

e. l'enregistrement "PARTIE_RECORD"

Cet enregistrement structure l'information au sujet d'une partie d'un formulaire.

Description de l'enregistrement:

```
DEFINE RECORD PARTIE_RECORD.
   PARTIE_REC STRUCTURE.
      NOM_PARTIE              DATATYPE TEXT            12.
      EMPLAC_PARTIE STRUCTURE OCCURS 10 TIMES.
         LIGNE_PART           DATATYPE UNSIGNED LONGWORD.
         COLONNE_PART         DATATYPE UNSIGNED LONGWORD.
         DUM_EMPLAC           DATATYPE TEXT             1.
      END EMPLAC_PARTIE STRUCTURE.
      P_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_PARTIE              DATATYPE TEXT       12.
         DUM_ET_PARTIE             DATATYPE TEXT        1.
      END P_ETATS_POSS STRUCTURE.
   END PARTIE_REC STRUCTURE.
END PARTIE_RECORD.
```

f. l'enregistrement "TEXTE_RECORD"

Cet enregistrement structure l'information au sujet d'un élément du squelette textuel d'un formulaire.

Description de l'enregistrement:

```
DEFINE RECORD TEXTE_RECORD.
   TEXTE_REC STRUCTURE.
      NOM_TEXTE               DATATYPE TEXT            12.
      FORMAT                  DATATYPE TEXT            30.
      CONTENU                 DATATYPE TEXT            30.
      EMPLAC_TEXTE STRUCTURE OCCURS 10 TIMES.
         LIGNE_TXT            DATATYPE UNSIGNED LONGWORD.
         COLONNE_TXT          DATATYPE UNSIGNED LONGWORD.
         DUM_TXT              DATATYPE TEXT             1.
      END EMPLAC_TEXTE STRUCTURE.
   END TEXTE_REC STRUCTURE.
END TEXTE_RECORD.
```

g. l'enregistrement "VARIABLE_RECORD"

Cet enregistrement structure l'information au sujet d'un élément variable d'un formulaire.

Description de l'enregistrement:

```
DEFINE RECORD VARIABLE_RECORD.
   VARIABLE_REC STRUCTURE.
      NOM_VAR                  DATATYPE TEXT           12.
      FORMAT                   DATATYPE TEXT           30.
      VAL_DEF                  DATATYPE TEXT           30.
      EMPLAC_VAR STRUCTURE OCCURS 10 TIMES.
         LIGNE_VAR             DATATYPE UNSIGNED LONGWORD.
         COLONNE_VAR           DATATYPE UNSIGNED LONGWORD.
         DUM_VAR               DATATYPE TEXT            1.
      END EMPLAC_VAR STRUCTURE.
      TXT_COR                  DATATYPE TEXT           12.
   END VARIABLE_REC STRUCTURE.
END VARIABLE_RECORD.
```

h. l'enregistrement "DOSSIER_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "dossier".

Description de l'enregistrement:

```
DEFINE RECORD DOSSIER_RECORD.
   DOSSIER_REC STRUCTURE.
      NOM_DOSSIER              DATATYPE TEXT      12.
      DO_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_DOSS            DATATYPE TEXT      12.
         DUM_ET_DOSS           DATATYPE TEXT       1.
      END DO_ETATS_POSS STRUCTURE.
      EXPED_DOSS               DATATYPE TEXT      12.
      DO_DESTINATAIRES STRUCTURE OCCURS 10 TIMES.
         DEST_DOSS             DATATYPE TEXT      12.
         DUM_DES_DOSS          DATATYPE TEXT       1.
      END DO_DESTINATAIRES STRUCTURE.
      CONSTITUE_DOSS STRUCTURE OCCURS 20 TIMES.
         CONST_DOSS            DATATYPE TEXT      12.
         TYPE_CONST_DOSS       DATATYPE TEXT      10.
         DUM_CONST             DATATYPE TEXT       1.
      END CONSTITUE_DOSS STRUCTURE.
   END DOSSIER_REC STRUCTURE.
END DOSSIER_RECORD.
```

i. l'enregistrement "FICHIER_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "fichier".

Description de l'enregistrement:

```
DEFINE RECORD FICHIER_RECORD.
   FICHIER_REC STRUCTURE.
      NOM_FICHIER              DATATYPE TEXT      12.
      FI_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_FICH            DATATYPE TEXT      12.
         DUM_ET_FICH           DATATYPE TEXT       1.
      END FI_ETATS_POSS STRUCTURE.
      CONSTITUE_FICH STRUCTURE.
         CONST_FICH            DATATYPE TEXT      12.
         TYPE_CONST_FICH       DATATYPE TEXT      10.
      END CONSTITUE_FICH STRUCTURE.
   END FICHIER_REC STRUCTURE.
END FICHIER_RECORD.
```

j. l'enregistrement "PILE_RECORD"

Cet enregistrement structure l'information au sujet de l'objet de bureau "pile".

Description de l'enregistrement:

```
DEFINE RECORD PILE_RECORD.
   PILE_REC STRUCTURE.
      NOM_PILE                 DATATYPE TEXT      12.
      PI_ETATS_POSS STRUCTURE OCCURS 10 TIMES.
         ETATS_PILE            DATATYPE TEXT      12.
         DUM_ET_PILE           DATATYPE TEXT       1.
      END PI_ETATS_POSS STRUCTURE.
   END PILE_REC STRUCTURE.
END PILE_RECORD.
```

C. LES REQUETES

Les requêtes définissent les échanges d'information entre le programme (définitions d'enregistrements) et le terminal (définitions d'écrans).

Dans chaque requête,
- On cite les définitions d'écran utilisées.
- On cite les définitions d'enregistrement utilisées.
- L'instruction "CLEAR SCREEN" permet d'obtenir un écran vierge avant tout traitement.
- L'opération "DISPLAY" permet d'afficher les écrans définis dans TDMS.
- L'instruction "INPUT" permet d'entrer une information saisie à partir d'une zone de l'écran dans une zone d'un enregistrement.
- L'instruction "OUTPUT" permet d'afficher une zone d'un enregistrement dans une zone d'un écran.
- Les clauses "PROGRAM KEY" et "CONTROL FIELD" permettent de gérer les erreurs.

a. la requête "BUREAU_INITIAL_REQUEST"

Cette requête permet d'afficher l'écran "BUREAU_INITIAL_FORM".

Description de la requête:

```
REPLACE BUREAU_INITIAL_REQUEST;

DESCRIPTION /* Cette requête affiche un texte descriptif.
               Elle explique comment utiliser le programme
               "BUREAU".
               Elle ne collecte aucune donnée.          */;

FORM IS BUREAU_INITIAL_FORM;

CLEAR SCREEN;
DISPLAY FORM BUREAU_INITIAL_FORM;

MESSAGE LINE IS "Tapez RETURN pour afficher le menu."

WAIT;

END DEFINITION;
```

b. la requête "BUREAU_MENU_REQUEST"

Cette requête permet d'afficher l'écran "BUREAU_MENU_FORM". Elle range dans l'enregistrement "BUREAU_TRAVAIL_RECORD" la sélection de l'utilisateur et le nom de l'objet qu'il désire décrire.

Description de la requête:

```
REPLACE REQUEST BUREAU_MENU_REQUEST;

DESCRIPTION  /*  Cette requête affiche un menu des choix de
                 l'application.
                 Elle collecte un numéro de sélection et le nom
                 de l'objet à ajouter.                    */;

FORM IS BUREAU_MENU_FORM;

RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM BUREAU_MENU_FORM;

   OUTPUT MESS_ERREUR TO MESS_ERREUR;
   INPUT SELECTION TO SELECTION;
   INPUT CONSULT_OBJET TO CONSULT_OBJET;

END DEFINITION;
```

c. la requête "AJOUT_MESSAGE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_MESSAGE_FORM". Elle range l'information concernant un message décrit par l'utilisateur dans l'enregistrement "MESSAGE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_MESSAGE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_MESSAGE_FORM;

RECORD IS MESSAGE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_MESSAGE_FORM;

   OUTPUT NOM_MESS TO NOM_MESS;
   INPUT TYPE_MESS TO TYPE_MESS;
   INPUT EXPLICATION_MESS TO EXPLICATION_MESS;
   INPUT ETATS_MESS[1 TO 10] TO ETATS_MESS[1 TO 10];
   INPUT EXPED_MESS TO EXPED_MESS;
   INPUT DEST_MESS[1 TO 10] TO DEST_MESS[1 TO 10];
   INPUT NOM[1 TO 10] TO NOM[1 TO 10];
   INPUT TYPE[1 TO 10] TO TYPE[1 TO 10];

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT TYPE_MESS TO TYPE_MESS;
 OUTPUT EXPLICATION_MESS TO EXPLICATION_MESS;
 OUTPUT ETATS_MESS[1 TO 10] TO ETATS_MESS[1 TO 10];
 OUTPUT EXPED_MESS TO EXPED_MESS;
 OUTPUT DEST_MESS[1 TO 10] TO DEST_MESS[1 TO 10];
 OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];
 OUTPUT TYPE[1 TO 10] TO TYPE[1 TO 10];
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

d. la requête "AJOUT_DOCUMENT_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_DOCUMENT_FORM". Elle range l'information concernant un document décrit par l'utilisateur dans l'enregistrement "DOCUMENT_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_DOCUMENT_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_DOCUMENT_FORM;

RECORD IS DOCUMENT_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_DOCUMENT_FORM;

   OUTPUT NOM_DOC TO NOM_DOC;
   INPUT TYPE_DOC TO TYPE_DOC;
   INPUT EXPLICATION_DOC TO EXPLICATION_DOC;
   INPUT ETATS_DOC[1 TO 10] TO ETATS_DOC[1 TO 10];
   INPUT EXPED_DOC TO EXPED_DOC;
   INPUT DEST_DOC[1 TO 10] TO DEST_DOC[1 TO 10];
   INPUT NOM[1 TO 10] TO NOM[1 TO 10];
   INPUT TYPE[1 TO 10] TO TYPE[1 TO 10];

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT TYPE_DOC TO TYPE_DOC;
 OUTPUT EXPLICATION_DOC TO EXPLICATION_DOC;
 OUTPUT ETATS_DOC[1 TO 10] TO ETATS_DOC[1 TO 10];
 OUTPUT EXPED_DOC TO EXPED_DOC;
 OUTPUT DEST_DOC[1 TO 10] TO DEST_DOC[1 TO 10];
 OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];
 OUTPUT TYPE[1 TO 10] TO TYPE[1 TO 10];
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

e. la requête "AJOUT_FORMULAIRE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_FORMULAIRE_FORM". Elle range l'information concernant un formulaire décrit par l'utilisateur dans l'enregistrement "FORMULAIRE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_FORMULAIRE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_FORMULAIRE_FORM;

RECORD IS FORMULAIRE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_FORMULAIRE_FORM;

   OUTPUT TITRE_FORM TO TITRE_FORM;
   INPUT TYPE_FORM TO TYPE_FORM;
   INPUT ETATS_FORM[1 TO 10] TO ETATS_FORM[1 TO 10];
   INPUT EXPED_FORM TO EXPED_FORM;
   INPUT DEST_FORM[1 TO 10] TO DEST_FORM[1 TO 10];
   INPUT NOM[1 TO 10] TO NOM[1 TO 10];
   INPUT TYPE[1 TO 10] TO TYPE[1 TO 10];

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT TYPE_FORM TO TYPE_FORM;
 OUTPUT ETATS_FORM[1 TO 10] TO ETATS_FORM[1 TO 10];
 OUTPUT EXPED_FORM TO EXPED_FORM;
 OUTPUT DEST_FORM[1 TO 10] TO DEST_FORM[1 TO 10];
 OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];
 OUTPUT TYPE[1 TO 10] TO TYPE[1 TO 10];
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

f. la requête "AJOUT_PARTIE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_PARTIE_FORM". Elle range l'information concernant une partie d'un formulaire décrit par l'utilisateur dans l'enregistrement "PARTIE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_PARTIE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_PARTIE_FORM;

RECORD IS PARTIE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_PARTIE_FORM;

   INPUT NOM_PARTIE TO NOM_PARTIE;
   INPUT LIGNE_PART[1 TO 10] TO LIGNE_PART[1 TO 10];
   INPUT COLONNE_PART[1 TO 10] TO COLONNE_PART[1 TO 10];
   INPUT ETATS_PARTIE[1 TO 10] TO ETATS_PARTIE[1 TO 10];
   INPUT SELECT_ELEMENT TO SELECT_ELEMENT;

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT NOM_PARTIE TO NOM_PARTIE;
 OUTPUT LIGNE_PART[1 TO 10] TO LIGNE_PART[1 TO 10];
 OUTPUT COLONNE_PART[1 TO 10] TO COLONNE_PART[1 TO 10];
 OUTPUT ETATS_PARTIE[1 TO 10] TO ETATS_PARTIE[1 TO 10];
 OUTPUT SELECT_ELEMENT TO SELECT_ELEMENT;
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

g. la requête "AJOUT_TEXTE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_TEXTE_FORM". Elle range l'information concernant un élément du squelette textuel d'un formulaire décrit par l'utilisateur dans l'enregistrement "TEXTE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_TEXTE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_TEXTE_FORM;

RECORD IS TEXTE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_TEXTE_FORM;

   INPUT NOM_TEXTE TO NOM_TEXTE;
   INPUT FORMAT TO FORMAT;
   INPUT CONTENU TO CONTENU;
   INPUT LIGNE_TXT[1 TO 10] TO LIGNE_TXT[1 TO 10];
   INPUT COLONNE_PART[1 TO 10] TO COLONNE_PART[1 TO 10];
   INPUT SELECT_ELEMENT TO SELECT_ELEMENT;

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT NOM_TEXTE TO NOM_TEXTE;
 OUTPUT FORMAT TO FORMAT;
 OUTPUT CONTENU TO CONTENU;
 OUTPUT LIGNE_TXT[1 TO 10] TO LIGNE_TXT[1 TO 10];
 OUTPUT COLONNE_TXT[1 TO 10] TO COLONNE_TXT[1 TO 10];
 OUTPUT SELECT_ELEMENT TO SELECT_ELEMENT;
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

h. la requête "AJOUT_VARIABLE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_VARIABLE_FORM". Elle range l'information concernant un élément variable d'un formulaire décrit par l'utilisateur dans l'enregistrement "VARIABLE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_VARIABLE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_VARIABLE_FORM;

RECORD IS VARIABLE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_VARIABLE_FORM;

   INPUT NOM_VAR TO NOM_VAR;
   INPUT FORMAT TO FORMAT;
   INPUT VAL_DEF TO VAL_DEF;
   INPUT LIGNE_VAR[1 TO 10] TO LIGNE_VAR[1 TO 10];
   INPUT COLONNE_VAR[1 TO 10] TO COLONNE_VAR[1 TO 10];
   INPUT TXT_COR TO TXT_COR;
   INPUT SELECT_ELEMENT TO SELECT_ELEMENT;

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT NOM_VAR TO NOM_VAR;
 OUTPUT FORMAT TO FORMAT;
 OUTPUT VAL_DEF TO VAL_DEF;
 OUTPUT LIGNE_VAR[1 TO 10] TO LIGNE_VAR[1 TO 10];
 OUTPUT COLONNE_VAR[1 TO 10] TO COLONNE_VAR[1 TO 10];
 OUTPUT TTX_COR TO TXT_COR;
 OUTPUT SELECT_ELEMENT TO SELECT_ELEMENT;
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

i. la requête "AJOUT_DOSSIER_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_DOSSIER_FORM". Elle range l'information concernant un dossier décrit par l'utilisateur dans l'enregistrement "DOSSIER_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_DOSSIER_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_DOSSIER_FORM;

RECORD IS DOSSIER_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_DOSSIER_FORM;

   OUTPUT NOM_DOSSIER TO NOM_DOSSIER;
   INPUT ETATS_DOSS[1 TO 10] TO ETATS_DOSS[1 TO 10];
   INPUT EXPED_DOSS TO EXPED_DOSS;
   INPUT DEST_DOSS[1 TO 10] TO DEST_DOSS[1 TO 10];
   INPUT CONST_DOSS[1 TO 20] TO CONST-DOSS[1 TO 20];
   INPUT TYPE_CONST_DOSS[1 TO 20] TO TYPE_CONST_DOSS[1 TO 20];

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT ETATS_DOSS[1 TO 10] TO ETATS_DOSS[1 TO 10];
 OUTPUT EXPED_DOSS TO EXPED_DOSS;
 OUTPUT DEST_DOSS[1 TO 10] TO DEST_DOSS[1 TO 10];
 OUTPUT CONST_DOSS[1 TO 20] TO CONST_DOSS[1 TO 20];
 OUTPUT TYPE_CONST_DOSS[1 TO 20] TO TYPE_CONST_DOSS[1 TO 20];
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

j. la requête "AJOUT_FICHIER_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_FICHIER_FORM". Elle range l'information concernant un fichier décrit par l'utilisateur dans l'enregistrement "FICHIER_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_FICHIER_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.          */;

FORM IS AJOUT_FICHIER_FORM;

RECORD IS FICHIER_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_FICHIER_FORM;

   OUTPUT NOM_FICHIER TO NOM_FICHIER;
   INPUT ETATS_FICH[1 TO 10] TO ETATS_FICH[1 TO 10];
   INPUT CONST_FICH TO CONST-FICH;
   INPUT TYPE_CONST_FICH TO TYPE_CONST_FICH;

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT ETATS_FICH[1 TO 10] TO ETATS_FICH[1 TO 10];
 OUTPUT CONST_FICH TO CONST_FICH;
 OUTPUT TYPE_CONST_FICH TO TYPE_CONST_FICH;
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

k. la requête "AJOUT_PILE_REQUEST";

Cette requête permet d'afficher l'écran "AJOUT_PILE_FORM". Elle range l'information concernant une pile décrite par l'utilisateur dans l'enregistrement "PILE_RECORD".

Description de la requête:

```
REPLACE REQUEST AJOUT_PILE_REQUEST;

DESCRIPTION  /*  Affiche les informations à fournir.
                 Collecte ces informations.            */;

FORM IS AJOUT_PILE_FORM;

RECORD IS PILE_RECORD;
RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM AJOUT_PILE_FORM;

   OUTPUT NOM_PILE TO NOM_PILE;
   INPUT ETATS_PILE[1 TO 10] TO ETATS_PILE[1 TO 10];

PROGRAM KEY IS GOLD "S"
   ! Si l'utilisateur tappe GOLD "S",
   ! le Save est retourné au programme.
   CHECK;
   RETURN "VERIFICATION" TO PROGRAM_REQUEST_KEY;
END PROGRAM KEY;

CONTROL FIELD IS VERIF_DONNEES
"Y":
 OUTPUT ETATS_PILE[1 TO 10] TO ETATS_PILE[1 TO 10];
END CONTROL FIELD;

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

END DEFINITION;
```

1. la construction de la librairie des requetes.

Toutes les requêtes décrites précédemment sont rangées dans le fichier de librairie des requêtes "BUREAULIB.RLB".

```
REPLACE LIBRARY BUREAU_LIBRARY;
   REQUEST IS BUREAU_INITIAL_REQUEST;
   REQUEST IS BUREAU_MENU_REQUEST;
   REQUEST IS AJOUT_MESSAGE_REQUEST;
   REQUEST IS AJOUT_MESSAGE_REQUSET;
   REQUEST IS AJOUT_DOCUMENT_REQUEST;
   REQUEST IS AJOUT_FORMULAIRE_REQUEST;
   REQUEST IS AJOUT_PARTIE_REQUEST;
   REQUEST IS AJOUT_TEXTE_REQUEST;
   REQUEST IS AJOUT_VARIABLE_REQUEST;
   REQUEST IS AJOUT_DOSSIER_REQUEST;
   REQUEST IS AJOUT_FICHIER_REQUEST;
   REQUEST IS AJOUT-PILE-REQUEST;
   FILE IS "BUREAULIB";
END DEFINITION;
```

2.3.2 LE PROGRAMME DE TRANSFORMATION EN C

On donnera d'abord la spécification du programme et de chaque fonction utilisée. On peut trouver ensuite le texte du programme dans le langage C.

A. SPECIFICATION

Pour chacune des fonctions, on donnera:
- une brève description en français de son effet;
- ses arguments: toutes les données nécessaires à l'exécution de la fonction;
- ses résultats;
- sa spécificaton sous forme de préconditions et postconditions;
- éventuellement, une description de son algorithme.

a. Le programme principal

* effet:
Ce programme a pour objectif de ranger dans un fichier une transformation en DSL des objets de bureau décrits au terminal par un utilisateur.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS"

* pré:
- le fichier "DEFINITIONS" est vide

* post:
- le fichier "DEFINITIONS" contient la description des objets de bureau sous forme de phrases DSL

* algorithme:
- ouverture de tous les fichiers nécessaires à la gestion d'écran au moyen de TDMS
  et ouverture de tous les fichiers de données (fichier "DEFINITIONS" qui contiendra la description des objets et fichier "OBJETS" qui contiendra le nom des objets décrits)
- description des concepts communs à tous les objets
- affichage de l'explication du programme
- tant que l'utilisateur désire ajouter des objets:
  - affichage du menu pour la sélection
  - appel de la fonction d'ajout d'un objet, correspondant à la sélection
- fermeture de tous les fichiers

b. La fonction TEST

* effet:
Cette fonction teste le Return Status renvoyé par les fonctions de TDMS.

* argument:
- RET_STAT: Return Status

* pré:
- il existe un Return Status RET_STAT

* post:
- si RET_STAT <> valeur normale, le programme est arrêté

c. La fonction EXIST

* effet:
Cette fonction teste si l'objet que l'on désire ajouter n'existe pas déjà.

* arguments:
- objet: nom de l'objet que l'on désire ajouter

* résultats:
- TROUVE: témoin de l'existence de l'objet

* pré:
- il existe un nom d'objet "objet"

* post:
- si "objet" existe déjà, alors TROUVE = vrai
sinon TROUVE = faux.

* algorithme:
- parcours du fichier "OBJETS" pour voir si l'objet existe déjà:
  - si l'objet courant dans le fichier "OBJETS" = l'objet que l'on désire ajouter:
    alors TROUVE = vrai et l'objet existe
    sinon on continue le parcours du fichier

d. La fonction CCOPY

* effet:
Cette fonction ajoute à une chaine de caractères fournie par TDMS le caractère "0" qui marque la fin d'une chaine de caractères en C.

* arguments:
- origine: chaine fournie par TDMS
- bmax: borne supérieure maximum de cette chaine

* résultats:
- copie: chaine traduite en C

* pré:
- il existe une chaine "origine"

* post:
- copie = origine + "0"

* algorithme:
- en commençant par la fin de la chaine offerte par TDMS et tant qu'on a à faire à un caractère blanc, on passe ce caractère
- on ajoute le caractère " " à la fin de la chaine

e. La fonction ADD_MESSAGE

* effet:
Cette fonction permet à l'utilisateur de décrire un message. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau message que l'utilisateur fournit

* algorithme:
  - on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
  - appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
  - les chaines de caractères en C devant se terminer par un caractère spécial ("0") et les chaines de caractères ne comprenant pas ce caractère, on est obligé de l'ajouter pour poursuivre la transformation des objets de bureau en DSL
    => traduction des enregistrements offerts par TDMS en structures C
  - ayant l'information voulue dans la structure C, on transforme l'objet décrit, en DSL, c'est-à-dire, on écrit sur le fichier "DEFINITIONS" les squelettes de texte DSL définis au chapitre 4 et complétés des renseignements fournis par l'utilisateur

f. La fonction ADD_DOCUMENT

* effet:
Cette fonction permet à l'utilisateur de décrire un document. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
  - fichier "DEFINITIONS"

* résultats:
  - fichier "DEFINITIONS'"

* pré:
  - le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
  - fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau document que l'utilisateur fournit

* algorithme:
  - on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
  - appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
  - traduction des enregistrements offerts par TDMS en structures C
  - transformation des renseignements fournis par l'utilisateur en phrases DSL

g. La fonction ADD_FORMULAIRE

* effet:
Cette fonction permet à l'utilisateur de décrire un formulaire. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau formulaire que l'utilisateur fournit

* algorithme:
- on dit à l'utilisateur comment décrire complètement un formulaire (description des généralités, parties, éléments du squelette textuel et éléments variables)
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation des généralités sur le formulaire en phrases DSL
- tant que la description du formulaire n'est pas complète:
  - selon que l'utilisateur désire ajouter une partie, un élément du squelette textuel ou un élément variable, appel à la fonction correspondante

h. La fonction ADD_PARTIE

* effet:
Cette fonction permet à l'utilisateur de décrire une partie d'un formulaire. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets et la description des généralités sur le formulaire auquel la partie appartient

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant la partie de formulaire que l'utilisateur fournit

* algorithme:
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation de la partie décrite, en DSL

i. La fonction ADD_TEXTE

* effet:
Cette fonction permet à l'utilisateur de décrire un élément du squelette textuel d'un formulaire. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets, la description des généralités sur le formulaire et la description de la partie à laquelle l'élément de texte appartient

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'élément de texte d'un formulaire que l'utilisateur fournit

* algorithme:
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation de l'élément de texte décrit, en DSL

j. La fonction ADD_VARIABLE

* effet:
Cette fonction permet à l'utilisateur de décrire un élément variable d'un formulaire. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets, la description des généralités sur le formulaire et la description de la partie à laquelle l'élément variable appartient

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'élément variable d'un formulaire que l'utilisateur fournit

* algorithme:
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation de l'élément variable décrit, en DSL

k. La fonction ADD_DOSSIER

* effet:
Cette fonction permet à l'utilisateur de décrire un dossier. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau dossier que l'utilisateur fournit

* algorithme:
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation du dossier décrit, en DSL

1. La fonction ADD_FICHIER

* effet:
  Cette fonction permet à l'utilisateur de décrire un fichier. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
  - fichier "DEFINITIONS"

* résultats:
  - fichier "DEFINITIONS'"

* pré:
  - le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
  - fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau fichier que l'utilisateur fournit

* algorithme:
  - on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
  - appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
  - traduction des enregistrements offerts par TDMS en structures C
  - transformation du fichier décrit, en DSL

m. La fonction ADD_PILE

* effet:
Cette fonction permet à l'utilisateur de décrire une pile. La description faite par l'utilisateur est ensuite transformée en phrases DSL.

* arguments:
- fichier "DEFINITIONS"

* résultats:
- fichier "DEFINITIONS'"

* pré:
- le fichier "DEFINITIONS" contient au moins les définitions communes à tous les objets

* post:
- fichier "DEFINITIONS'" = fichier "DEFINITIONS" + phrases DSL décrivant l'objet de bureau pile que l'utilisateur fournit

* algorithme:
- on initialise avec des caractères blancs l'enregistrement qui va servir à ranger les informations données par l'utilisateur
- appel de la requête TDMS qui va permettre à l'utilisateur de décrire son objet
- traduction des enregistrements offerts par TDMS en structures C
- transformation de la pile décrite, en DSL

B. TEXTE DU PROGRAMME

```
/*****************************************************************/
/*****************************************************************/
/*************                                        ************/
/*************        PROGRAMME  AJOUT_BUREAU         ************/
/*************                                        ************/
/*****************************************************************/
/*****************************************************************/
/***     Ce programme collecte les descriptions des objets     ***/
/***     de bureau et les traduit en DSL.                      ***/
/*****************************************************************/
/*****************************************************************/


#include stdio      /* définitions relatives aux input/output */

#include descrip    /* définitions relatives aux appels par
                        descripteurs */

/*  inclusion des records definis dans TDMS  */

#dictionary "BUREAU_TRAVAIL_RECORD"
#dictionary "MESSAGE_RECORD"
#dictionary "FORMULAIRE_RECORD"
#dictionary "PARTIE_RECORD"
#dictionary "TEXTE_RECORD"
#dictionary "VARIABLE_RECORD"
#dictionary "DOCUMENT_RECORD"
#dictionary "DOSSIER_RECORD"
#dictionary "FICHIER_RECORD"
#dictionary "PILE_RECORD"


/*  déclaration des constantes  */

#define  TRUE      1
#define  FALSE     0


/*  déclaration des variables  */

int    RET_STAT;   /* Return Status des fonctions de TDMS */
int    CHANNEL;    /* canal vers le terminal */
int    LIBRARY_ID; /* identificateur de la librairie de TDMS */
int    BON_SELEC;  /* témoin de bonne sélection: booleen */
int    T_ERR;      /* test de la bonne sélection des objets */
```

```
/*  déclaration des enregistrements de saisie dans TDMS  */

struct   bureau_travail_rec    trav_bur;
struct   message_rec    msge;
struct   document_rec    doct;
struct   formulaire_rec    form;
struct   partie_rec    part;
struct   texte_rec    elt_txt;
struct   variable_rec    elt_var;
struct   dossier_rec   doss;
struct   fichier_rec   fich;
struct   pile_rec    tas;



/*  déclaration des structures utilisées en c  */

struct   c_bureau_travail
{
    int   c_select;
    char  c_mess_err[81];
    char  c_consult_objet[13];
    char  c_prog_req_key[13];
    char  c_verif_donnees;
    char  c_select_elt;
}   c_trav_bur;


struct   c_message
{
    char  c_nom_mess[13];
    char  c_type_mess[13];
    char  c_explication_mess[13];
    struct
    {
        char  c_etats_mess[13];
        char  c_dum_etat;
    }   c_m_etats_poss[10];
    char  c_exped_mess[13];
    struct
    {
        char  c_dest_mess[13];
        char  c_dum_dest;
    }   c_m_destinataires[10];
    struct
    {
        char  c_nom[13];
        char  c_type[11];
        char  c_dum_rep;
    }   c_reponse_mess[10];
}   c_msge;
```

```
struct   c_document
{
    char  c_nom_doc[13];
    char  c_type_doc[13];
    char  c_explication_doc[13];
    struct
    {
        char  c_etats_doc[13];
        char  c_dum_etat;
    }   c_d_etats_poss[10];
    char  c_exped_doc[13];
    struct
    {
        char  c_dest_doc[13];
        char  c_dum_dest;
    }   c_d_destinataires[10];
    struct
    {
        char  c_nom[13];
        char  c_type[11];
        char  c_dum_rep;
    }   c_reponse_doc[10];
}   c_doct;
```

```
struct   c_formulaire
{
    char  c_titre_form[13];
    char  c_type_form[11];
    struct
    {
        char  c_etats_form[13];
        char  c_dum_et_form;
    }   c_f_etats_poss[10];
    char  c_exped_form[13];
    struct
    {
        char  c_dest_form[13];
        char  c_dum_des_form;
    }   c_f_destinataires[10];
    struct
    {
        char  c_nom[13];
        char  c_type[11];
        char  c_dum_rep;
    }   c_reponse_form[10];
}   c_form;
```

```
struct    c_partie
{
    char   c_nom_partie[13];
    struct
    {
        int    c_ligne_part;
        int    c_colonne_part;
        char   c_dum_emplac;
    }   c_emplac_partie[10];
    struct
    {
        char   c_etats_partie[13];
        char   c_dum_et_partie;
    }   c_p_etats_poss[10];
}   c_part;


struct   c_texte
{
    char   c_nom_texte[13];
    char   c_format[31];
    char   c_contenu[31];
    struct
    {
        int    c_ligne_txt;
        int    c_colonne_txt;
            char   c_dum_txt;
    }   c_emplac_texte[10];
}     c_elt_txt;



struct    c_variable
{
    char   c_nom_var[13];
    char   c_format[31];
    char   c_val_def[31];
    struct
    {
        int    c_ligne_var;
        int    c_colonne_var;
        char   c_dum_var;
    }   c_emplac_var[10];
    char   c_txt_cor[13];
}   c_elt_var;
```

```
struct   c_dossier
{
    char  c_nom_dossier[13];
    struct
    {
        char  c_etats_doss[13];
        char  c_dum_et_doss;
    }   c_do_etats_poss[10];
    char  c_exped_doss[13];
    struct
    {
        char  c_dest_doss[13];
        char  c_dum_des_doss;
    }   c_do_destinataires[10];
    struct
    {
        char  c_const_doss[13];
        char  c_type_const_doss[11];
        char  c_dum_const;
    }   c_constitue_doss[20];
}   c_doss;
```

```
struct   c_fichier
{
    char  c_nom_fichier[13];
    struct
    {
        char  c_etats_fich[13];
        char  c_dum_et_fich;
    }   c_fi_etats_poss[10];
    struct
    {
        char  c_const_fich[13];
        char  c_type_const_fich[11];
    }   c_constitue_fich;
}   c_fich;
```

```
struct   c_pile
{
    char  c_nom_pile[13];
    struct
    {
        char  c_etats_pile[13];
        char  c_dum_et_pile;
    }   c_pi_etats_poss[10];
}   c_tas;
```

```
/*  déclaration des fichiers  */

FILE      *fopen( );
FILE      *OBJETS;
FILE      *DEFINITIONS;

/*  déclaration des fonctions de TDMS  */

extern    int    TSS$OPEN( );
extern    int    TSS$OPEN_RLB( );
extern    int    TSS$REQUEST( );
extern    int    TSS$CLOSE( );
extern    int    TSS$CLOSE_RLB( );
extern    int    LIB$STOP( );


/* déclaration des descripteurs de la librairie et des requêtes */

$DESCRIPTOR(LIB_DESC,"BUREAULIB.RLB");
$DESCRIPTOR(INIT_DESC,"BUREAU_INITIAL_REQUEST");
$DESCRIPTOR(MENU_DESC,"BUREAU_MENU_REQUEST");
$DESCRIPTOR(MESS_DESC,"AJOUT_MESSAGE_REQUEST");
$DESCRIPTOR(DOSS_DESC,"AJOUT_DOSSIER_REQUEST");
$DESCRIPTOR(DOC_DESC,"AJOUT_DOCUMENT_REQUEST");
$DESCRIPTOR(FICH_DESC,"AJOUT_FICHIER_REQUEST");
$DESCRIPTOR(PILE_DESC,"AJOUT_PILE_REQUEST");
$DESCRIPTOR(EXP_DESC,"FORMULAIRE_EXPLICATION_REQUEST");
$DESCRIPTOR(FORM_DESC,"AJOUT_FORMULAIRE_REQUEST");
$DESCRIPTOR(PART_DESC,"AJOUT_PARTIE_REQUEST");
$DESCRIPTOR(TEXTE_DESC,"AJOUT_TEXTE_REQUEST");
$DESCRIPTOR(VAR_DESC,"AJOUT_VARIABLE_REQUEST");
```

```
/*****************************************************************/
/****                    PROGRAMME PRINCIPAL                  ****/
/*****************************************************************/


main( )
{
   /*  ouverture des fichiers de données  */

   if ((OBJETS = fopen("DEFOBJETS.DTA","w")) == NULL)
   {
      printf("erreur:fichier non ouvert\n");
      exit(RET_STAT);
   }
   else
   {
      fclose(OBJETS);
      if (( OBJETS = fopen("DEFOBJETS.DTA","w+")) == NULL)
      {
          exit(RET_STAT);
      }
   }
   if ((DEFINITIONS = fopen("DEFBUREAU.DSL","w")) == NULL)
   {
      printf("erreur:fichier non ouvert\n");
      exit(RET_STAT);
   }
   else
   {
      fclose(DEFINITIONS);
      if (( DEFINITIONS = fopen("DEFBUREAU.DSL","w+")) == NULL)
      {
          exit(RET_STAT);
      }
   }


   /*  ouverture de la librairie de requêtes  */

   RET_STAT = TSS$OPEN_RLB(&LIB_DESC,&LIBRARY_ID);
   test(RET_STAT);

   /*  ouverture du canal vers le terminal */

   RET_STAT = TSS$OPEN(&CHANNEL);
   test(RET_STAT);

   /*  traduction des concepts communs à tous les objets  */

   fprintf(DEFINITIONS,"DEFINE  ELEMENT  contenu;\n");
   fprintf(DEFINITIONS,"   FORMAT IS  \"texte\";\n");
   fprintf(DEFINITIONS,"DEFINE  ELEMENT  echeance;\n");
   fprintf(DEFINITIONS,"   FORMAT IS  \"99/99/99\";\n");
   fprintf(DEFINITIONS,"DEFINE  ELEMENT  status;\n");
   fprintf(DEFINITIONS,"   DOMAIN OF VALUE ARE original,copie;\n");
```

```
fprintf(DEFINITIONS,"DEFINE  ELEMENT  support;\n");
fprintf(DEFINITIONS,"   FORMAT IS  \"alphabetique\";\n");
fprintf(DEFINITIONS,"DEFINE  GROUP sujet;\n");
fprintf(DEFINITIONS,"   CONSISTS OF  5 mot_cle;\n");
fprintf(DEFINITIONS,"DEFINE  ELEMENT  mot_cle;\n");
fprintf(DEFINITIONS,"   FORMAT IS  \"alphabetique\";\n");
fprintf(DEFINITIONS,"DEFINE  ELEMENT  corps;\n");
fprintf(DEFINITIONS,"   FORMAT IS  \"texte\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT paragraphes_statiques;\n");
fprintf(DEFINITIONS,"   FORMAT IS  \"booleen\";\n");



/*  affichage de l'explication du programme  */

RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&INIT_DESC);
test(RET_STAT);




/*  ajout des objets de bureau  */

trav_bur.selection = 0;
while (trav_bur.selection != 2)
        /* tant que je veux ajouter des objets */
{
   strncpy(trav_bur.mess_erreur,"        ",80);
   BON_SELEC = FALSE;
   while (BON_SELEC == FALSE)
   {
      /*  affichage de la sélection  */

      trav_bur.selection = 0;
      strncpy(trav_bur.consult_objet,"            ",12);
      RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&MENU_DESC,
                             &trav_bur);
      test(RET_STAT);


      /* traduction de la structure TDMS en une structure C */

      c_trav_bur.c_select = trav_bur.selection;
      ccopy(c_trav_bur.c_mess_err,trav_bur.mess_erreur,80);
      ccopy(c_trav_bur.c_consult_objet,
            trav_bur.consult_objet,12);
      ccopy(c_trav_bur.c_prog_req_key,
            trav_bur.program_request_key,12);
      c_trav_bur.c_verif_donnees = trav_bur.verif_donnees;
      c_trav_bur.c_select_elt = trav_bur.select_element;
```

```
      /*  test des paramètres rentrés lors de la sélection  */

      T_ERR = 0;
      if ((c_trav_bur.c_select != 2)
         && (c_trav_bur.c_select != 11)
         && (c_trav_bur.c_select != 12)
         && (c_trav_bur.c_select != 13)
         && (c_trav_bur.c_select != 14)
         && (c_trav_bur.c_select != 15)
         && (c_trav_bur.c_select != 16))
               T_ERR = 1;
      if (c_trav_bur.c_select != 2)
      {
         if ((exist(c_trav_bur.c_consult_objet)) == TRUE)
               T_ERR += 2;
      }

      switch (T_ERR)
      {
         case 0: BON_SELEC = TRUE;
                 fprintf(OBJETS,"%s/n",
                              c_trav_bur.c_consult_objet);
                 break;

         case 1: strncpy(trav_bur.mess_erreur,"L'operation
                              choisie n'existe pas.",80);
                 break;

         case 2: strncpy(trav_bur.mess_erreur,"L'objet a
                              ajouter existe deja.",80);
                 break;

         case 3: strncpy(trav_bur.mess_erreur,"L'operation
                        choisie n'existe pas  e t l'objet
                                          existe deja",80);
                 break;

      }     /*  fin du switch sur T_ERR  */


   }        /*  fin du while (BON_SELEC == FALSE)  */


   /*  selon la sélection, appel aux fonctions de saisie
       d'information sur les objets et traduction en DSL  */


   switch (trav_bur.selection)
     {
        case 11: add_message( );
                 break;

        case 12: add_document( );
                 break;
```

```
            case 13: add_formulaire( );
                     break;

            case 14: add_dossier( );
                     break;

            case 15: add_fichier( );
                     break;

            case 16: add_pile( );
                     break;

            case 2: break;

          }       /*  fin du switch sur SELECTION  */

        }         /*  fin du while (SELECTION != 2)  */



        /*  refermer tous les fichiers  */

        fclose(OBJETS);
        fclose(DEFINITIONS);
        RET_STAT = TSS$CLOSE_RLB(&LIBRARY_ID);
        test(RET_STAT);
        RET_STAT = TSS$CLOSE(&CHANNEL);
        test(RET_STAT);


    }    /*  fin du programme principal  */



/*****************************************************************/
```

```
/*****************************************************************/
/*****************************************************************/
/*****                      FONCTIONS                        *****/
/*****************************************************************/
/*****************************************************************/




/*****************************************************************/
/*****                   FONCTION CCOPY                      *****/
/*****************************************************************/

ccopy(copie,origine,bmax)
   char *origine;
   char *copie;
   int  bmax;
{
   int   i;

    i = bmax - 1;
    while (( origine[i] == ' ' )
          && ( i >= 0 ))
    {
       i = i - 1;
    }
    copie[i + 1] = '\0';
    while ( i >= 0 )
    {
       copie[i] = origine[i];
       i = i - 1;
    }
}



/*****************************************************************/
/*****                   FONCTION  TEST                      *****/
/*****************************************************************/


test(RET_STAT)
{
   if (RET_STAT != 2981889)    /* SI RET_STAT != TSS$NORMAL   */
         LIB$STOP(RET_STAT);
}
```

```
/*****************************************************************/
/*****                    FONCTION EXIST                     *****/
/*****************************************************************/


int  exist(objet)
   char   objet[13];

   {
       int   TROUVE = FALSE;
       char  buffer[14];
       int   i;

       fseek(OBJETS,0,0);
       while (fgets(buffer,14,OBJETS) != NULL)
         {
            for (i = 0;i < 14;i++)
            {
               if (buffer[i] == '/')
               {
                  buffer[i] = '\0';
                  break;
               }
            }

            if (strcmp(buffer,objet) == 0)
                 TROUVE = TRUE;
         }
       return (TROUVE);

    }
```

```
/*****************************************************************/
/*****                FONCTION  ADD_MESSAGE                  *****/
/*****************************************************************/


add_message( )
{
   /*  déclaration des variables internes  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  compteur: nombre d'etats  */
   int   nb_rep;     /*  compteur: nombre de reponses  */
   int   nb_dest;    /*  compteur: nombre de destinataires  */
```

```
/*  initialisation de la structure qui contiendra les
    informations  */

strncpy(msge.nom_mess,trav_bur.consult_objet,12);
strncpy(msge.type_mess,"            ",12);
strncpy(msge.explication_mess,"              ",12);
strncpy(msge.exped_mess,"            ",12);
for (i = 0; i < 10; i++)
{
  strncpy(msge.m_etats_poss[i].etats_mess,
                          "              ",12);
  msge.m_etats_poss[i].dum_etat = ' ';
  strncpy(msge.m_destinataires[i].dest_mess,
                          "              ",12);
  msge.m_destinataires[i].dum_dest = ' ';

  strncpy(msge.reponse_mess[i].nom,"             ",12);
  strncpy(msge.reponse_mess[i].type,"           ",10);
  msge.reponse_mess[i].dum_rep = ' ';
   }


/*  appel de la requête d'affichage du message  */

strncpy(trav_bur.program_request_key,"             ",12);
trav_bur.verif_donnees = 'N';
RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&MESS_DESC,
                       &msge, &trav_bur);
test(RET_STAT);

ccopy(c_trav_bur.c_prog_req_key,
                trav_bur.program_request_key,12);
while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0")
                                                  == 0)
{
  trav_bur.verif_donnees = 'Y';
  strncpy(trav_bur.program_request_key,"             ",12);
  RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&MESS_DESC,&msge,
                         &trav_bur);
  test(RET_STAT);
  ccopy(c_trav_bur.c_prog_req_key,
                    trav_bur.program_request_key,12);
}


/*  traduction de la structure TDMS en une structure C  */

ccopy(c_msge.c_nom_mess,trav_bur.consult_objet,12);
ccopy(c_msge.c_type_mess,msge.type_mess,12);
ccopy(c_msge.c_explication_mess,msge.explication_mess,12);
ccopy(c_msge.c_exped_mess,msge.exped_mess,12);
```

```
for (i = 0; i < 10; i++)
  {
     ccopy(c_msge.c_m_etats_poss[i].c_etats_mess,
                      msge.m_etats_poss[i].etats_mess,12);
     c_msge.c_m_etats_poss[i].c_dum_etat
                          = msge.m_etats_poss[i].dum_etat;
     ccopy(c_msge.c_m_destinataires[i].c_dest_mess,
                    msge.m_destinataires[i].dest_mess,12);
     c_msge.c_m_destinataires[i].c_dum_dest
                       = msge.m_destinataires[i].dum_dest;
     ccopy(c_msge.c_reponse_mess[i].c_nom,
                               msge.reponse_mess[i].nom,12);
     ccopy(c_msge.c_reponse_mess[i].c_type,
                              msge.reponse_mess[i].type,10);
     c_msge.c_reponse_mess[i].c_dum_rep
                              = msge.reponse_mess[i].dum_rep;
   }



/*  L'information voulue est dans la structure MSGE.  */
/*  Je traduis en DSL.  */


/*  Description de l'objet DSL "ENTITY" lié au message.  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                        c_msge.c_nom_mess);
fprintf(DEFINITIONS,"  KEYWORD ARE  \"MSGE\";\n");
fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                        c_msge.c_nom_mess);
fprintf(DEFINITIONS,"               explication_%s,\n",
                                        c_msge.c_nom_mess);
fprintf(DEFINITIONS,"               contenu,\n");
fprintf(DEFINITIONS,"               etat_%s,\n",
                                        c_msge.c_nom_mess);
fprintf(DEFINITIONS,"               echeance,\n");
fprintf(DEFINITIONS,"               status,\n");
fprintf(DEFINITIONS,"               support;\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  type_%s;\n",
                                      c_msge.c_nom_mess);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                                    c_msge.c_type_mess);
fprintf(DEFINITIONS,"DEFINE ELEMENT  explication_%s;\n",
                                      c_msge.c_nom_mess);
if (c_msge.c_explication_mess[0] != '\0')
{
    fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                                  c_msge.c_explication_mess);
}
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                      c_msge.c_nom_mess);
```

```
if (c_msge.c_m_etats_poss[1].c_etats_mess[0] == '\0')
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                      c_msge.c_m_etats_poss[0].c_etats_mess);
}
else
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                     c_msge.c_m_etats_poss[0].c_etats_mess);
}
nb_et = 1;
while ((c_msge.c_m_etats_poss[nb_et].c_etats_mess[0] != '\0')
       && (nb_et < 10))
{
   if (c_msge.c_m_etats_poss[nb_et + 1].c_etats_mess[0] == '\0')
     {
        fprintf(DEFINITIONS,"                              %s;\n",
                c_msge.c_m_etats_poss[nb_et].c_etats_mess);
     }
   else
    {
        fprintf(DEFINITIONS,"                              %s,\n",
                c_msge.c_m_etats_poss[nb_et].c_etats_mess);
     }
   nb_et = nb_et + 1;

}



/* traduction des objets DSL "MESSAGE" liés au message
   du modèle proposé */

nb_et = 0;
while ((c_msge.c_m_etats_poss[nb_et].c_etats_mess[0] != '\0')
       && (nb_et < 10))
{
   fprintf(DEFINITIONS,"DEFINE MESSAGE  %s_%s_MESS;\n",
c_msge.c_nom_mess,c_msge.c_m_etats_poss[nb_et].c_etats_mess );
   fprintf(DEFINITIONS,"  KEYWORD ARE  \"MSGE\";\n");
   fprintf(DEFINITIONS,"  ATTRIBUTE IS  associe_a   \"%s_ENT\";\n",
                                        c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                        c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"               explication_%s,\n",
                                        c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"               contenu,\n");
   fprintf(DEFINITIONS,"               echeance,\n");
   fprintf(DEFINITIONS,"               status,\n");
   fprintf(DEFINITIONS,"               support;\n");
   nb_et = nb_et + 1;
}
```

```
   /*  traduction des réponses attendues  */

   nb_rep = 0;
   while ((nb_rep < 10)
         && (c_msge.c_reponse_mess[nb_rep].c_nom[0] != '\0'))
   {
      fprintf(DEFINITIONS,"DEFINE RELATION  %s_REP_%s;\n",
          c_msge.c_nom_mess,c_msge.c_reponse_mess[nb_rep].c_nom);
      fprintf(DEFINITIONS,"  RELATES %s_ENT\n",c_msge.c_nom_mess);
      fprintf(DEFINITIONS,"    AS  a_pour_reponse\n");
      fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
      fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                          c_msge.c_reponse_mess[nb_rep].c_nom);
      fprintf(DEFINITIONS,"    AS  repond_a\n");
      fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
      nb_rep = nb_rep + 1;
   }

   /*  traduction de l'expéditeur du message  */

   fprintf(DEFINITIONS,"DEFINE RELATION  %s_EXP_%s;\n",
                       c_msge.c_nom_mess,c_msge.c_exped_mess);
   fprintf(DEFINITIONS,"  CONSISTS OF  date_exp_%s;\n",
                                       c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"    AS  expedie_par\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_msge.c_exped_mess);
   fprintf(DEFINITIONS,"    AS  expedie\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   fprintf(DEFINITIONS,"DEFINE ELEMENT  date_exp_%s;\n",
                                       c_msge.c_nom_mess);
   fprintf(DEFINITIONS,"  FORMAT IS  \"99/99/9999\";\n");

   /*  traduction des destinataires du message  */

   nb_dest = 0;
   while ((nb_dest < 10)
       &&(c_msge.c_m_destinataires[nb_dest].c_dest_mess[0]
                                                   != '\0'))
   {
     fprintf(DEFINITIONS,"DEFINE RELATION  %s_DEST_%s;\n",
                 c_msge.c_nom_mess,
                 c_msge.c_m_destinataires[nb_dest].c_dest_mess);
     fprintf(DEFINITIONS," RELATES  %s_ENT\n",c_msge.c_nom_mess);
     fprintf(DEFINITIONS," AS destine_a\n");
     fprintf(DEFINITIONS," WITH CONNECTIVITY \"0_N\";\n");
     fprintf(DEFINITIONS," RELATES  %s_ENT\n",
                 c_msge.c_m_destinataires[nb_dest].c_dest_mess);
     fprintf(DEFINITIONS," AS recoit\n");
     fprintf(DEFINITIONS," WITH CONNECTIVITY \"0_N\";\n");
     nb_dest = nb_dest + 1; }

}
/*******  fin de la traduction du message          **********/
```

```
/*****************************************************************/
/*****                  FONCTION  ADD_DOCUMENT                *****/
/*****************************************************************/


add_document( )
{
   /*  declaration des variables internes  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  compteur: nombre d'etats  */
   int   nb_rep;     /*  compteur: nombre de reponses  */
   int   nb_dest;    /*  compteur: nombre de destinataires  */


   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(doct.nom_doc,trav_bur.consult_objet,12);
   strncpy(doct.type_doc,"             ",12);
   strncpy(doct.explication_doc,"              ",12);
   strncpy(doct.exped_doc,"            ",12);
   for (i = 0; i < 10; i++)
   {
     strncpy(doct.d_etats_poss[i].etats_doc,"             ",12);
     doct.d_etats_poss[i].dum_etat = ' ';
     strncpy(doct.d_destinataires[i].dest_doc,"             ",12);
     doct.d_destinataires[i].dum_dest = ' ';
     strncpy(doct.reponse_doc[i].nom,"            ",12);
     strncpy(doct.reponse_doc[i].type,"          ",10);
     doct.reponse_doc[i].dum_rep = ' ';
    }
    /*  appel de la requête d'affichage du document  */

    strncpy(trav_bur.program_request_key,"            ",12);
    trav_bur.verif_donnees = 'N';
    RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOC_DESC,
                           &doct,&trav_bur);
    test(RET_STAT);

    ccopy(c_trav_bur.c_prog_req_key,
                          trav_bur.program_request_key,12);
    while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0")
                                                    == 0)
      {
        trav_bur.verif_donnees = 'Y';
        strncpy(trav_bur.program_request_key,"            ",12);
        RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOC_DESC,
                               &doct,&trav_bur);
        test(RET_STAT);
        ccopy(c_trav_bur.c_prog_req_key,
                          trav_bur.program_request_key,12);
      }
```

```
/*  traduction de la structure TDMS en une structure C  */

ccopy(c_doct.c_nom_doc,trav_bur.consult_objet,12);
ccopy(c_doct.c_type_doc,doct.type_doc,12);
ccopy(c_doct.c_explication_doc,doct.explication_doc,12);
ccopy(c_doct.c_exped_doc,doct.exped_doc,12);
for (i = 0; i < 10; i++)
  {
     ccopy(c_doct.c_d_etats_poss[i].c_etats_doc,
                    doct.d_etats_poss[i].etats_doc,12);
     c_doct.c_d_etats_poss[i].c_dum_etat
                      = doct.d_etats_poss[i].dum_etat;
     ccopy(c_doct.c_d_destinataires[i].c_dest_doc,
                  doct.d_destinataires[i].dest_doc,12);
     c_doct.c_d_destinataires[i].c_dum_dest
                    = doct.d_destinataires[i].dum_dest;
     ccopy(c_doct.c_reponse_doc[i].c_nom,
                         doct.reponse_doc[i].nom,12);
     ccopy(c_doct.c_reponse_doc[i].c_type,
                        doct.reponse_doc[i].type,12);
     c_doct.c_reponse_doc[i].c_dum_rep
                       = doct.reponse_doc[i].dum_rep;
   }


/*  L'information voulue est dans la structure DOCT.  */
/*  Je traduis en DSL.  */


/*  Description de l'objet DSL "ENTITY" lié au document.  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_doct.c_nom_doc);
fprintf(DEFINITIONS,"  KEYWORD ARE  \"DOCUMENT\";\n");
fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                    c_doct.c_nom_doc);
fprintf(DEFINITIONS,"               explication_%s,\n",
                                          c_doct.c_nom_doc);
fprintf(DEFINITIONS,"               contenu,\n");
fprintf(DEFINITIONS,"               etat_%s,\n",
                                          c_doct.c_nom_doc);
fprintf(DEFINITIONS,"               echeance,\n");
fprintf(DEFINITIONS,"               status,\n");
fprintf(DEFINITIONS,"               support;\n");
```

```
fprintf(DEFINITIONS,"DEFINE ELEMENT  type_%s;\n",
                                        c_doct.c_nom_doc);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                                       c_doct.c_type_doc);
fprintf(DEFINITIONS,"DEFINE ELEMENT  explication_%s;\n",
                                        c_doct.c_nom_doc);
if (c_doct.c_explication_doc[0] != '\0')
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                                    c_doct.c_explication_doc);
}
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                       c_doct.c_nom_doc);
if (c_doct.c_d_etats_poss[1].c_etats_doc[0] == '\0')
{
   fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                        c_doct.c_d_etats_poss[0].c_etats_doc);
}
else
{
   fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                       c_doct.c_d_etats_poss[0].c_etats_doc);
}
nb_et = 1;
while ((c_doct.c_d_etats_poss[nb_et].c_etats_doc[0] != '\0')
       && (nb_et < 10))
{
   if (c_doct.c_d_etats_poss[nb_et + 1].c_etats_doc[0] == '\0')
     {
        fprintf(DEFINITIONS,"                          %s;\n",
                  c_doct.c_d_etats_poss[nb_et].c_etats_doc);
     }
   else
     {
        fprintf(DEFINITIONS,"                          %s,\n",
                c_doct.c_d_etats_poss[nb_et].c_etats_doc);
     }
   nb_et = nb_et + 1;

}



/* traduction des objets DSL "MESSAGE" liés au document
   du modèle proposé */

nb_et = 0;
while ((c_doct.c_d_etats_poss[nb_et].c_etats_doc[0] != '\0')
       && (nb_et < 10))
{
   fprintf(DEFINITIONS,"DEFINE MESSAGE  %s_%s_MESS;\n",
 c_doct.c_nom_doc,c_doct.c_d_etats_poss[nb_et].c_etats_doc);
   fprintf(DEFINITIONS,"  KEYWORD ARE  \"DOCUMENT\";\n");
   fprintf(DEFINITIONS,"  ATTRIBUTE IS  associe_a   \"%s_ENT\";\n",
                                      c_doct.c_nom_doc);
```

```
   fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                        c_doct.c_nom_doc);
   fprintf(DEFINITIONS,"               explication_%s,\n",
                                        c_doct.c_nom_doc);
   fprintf(DEFINITIONS,"               contenu,\n");
   fprintf(DEFINITIONS,"               echeance,\n");
   fprintf(DEFINITIONS,"               status,\n");
   fprintf(DEFINITIONS,"               support;\n");
   nb_et = nb_et + 1;
}


/*  traduction des réponses attendues  */

nb_rep = 0;
while ((nb_rep < 10)
       && (c_doct.c_reponse_doc[nb_rep].c_nom[0] != '\0'))
{
   fprintf(DEFINITIONS,"DEFINE RELATION  %s_REP_%s;\n",
     c_doct.c_nom_doc,c_doct.c_reponse_doc[nb_rep].c_nom);
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_doct.c_nom_doc);
   fprintf(DEFINITIONS,"    AS  a_pour_reponse\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                        c_doct.c_reponse_doc[nb_rep].c_nom);
   fprintf(DEFINITIONS,"    AS  repond_a\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   nb_rep = nb_rep + 1;
}


/*  traduction de l'expéditeur du document  */

fprintf(DEFINITIONS,"DEFINE RELATION  %s_EXP_%s;\n",
                      c_doct.c_nom_doc,c_doct.c_exped_doc);
fprintf(DEFINITIONS,"  CONSISTS OF  date_exp_%s;\n",
                                      c_doct.c_nom_doc);
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_doct.c_nom_doc);
fprintf(DEFINITIONS,"    AS  expedie_par\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_doct.c_exped_doc);
fprintf(DEFINITIONS,"    AS  expedie\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  date_exp_%s;\n",
                                     c_doct.c_nom_doc);
fprintf(DEFINITIONS,"  FORMAT IS  \"99/99/9999\";\n");
```

```
   /*  traduction des destinataires du document  */

   nb_dest = 0;
   while ((nb_dest < 10)
    && (c_doct.c_d_destinataires[nb_dest].c_dest_doc[0] != '\0'))
     {
       fprintf(DEFINITIONS,"DEFINE RELATION  %s_DEST_%s;\n",
                c_doct.c_nom_doc,
                c_doct.c_d_destinataires[nb_dest].c_dest_doc);
       fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_doct.c_nom_doc);
       fprintf(DEFINITIONS,"    AS  destine_a\n");
       fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
       fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                c_doct.c_d_destinataires[nb_dest].c_dest_doc);
       fprintf(DEFINITIONS,"    AS  recoit\n");
       fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
       nb_dest = nb_dest + 1;
     }

}

/*****  fin de la traduction du document          *****/



/*****************************************************************/
/*****          FONCTION  ADD_FORMULAIRE                    *****/
/*****************************************************************/


add_formulaire( )
{
   /*  variables  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  nombre d'états  */
   int   nb_rep;     /*  nombre de réponses  */
   int   nb_dest;    /*  nombre de destinataires  */
   int   choix;



   /*  affichage de l'explication  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&EXP_DESC);
   test(RET_STAT);
 .BP
   /*  saisie des généralites sur le formulaire  */

   /*  initialisation de la structure FORM  */

   strncpy(form.titre_form,trav_bur.consult_objet,12);
   strncpy(form.type_form,"          ",10);
   strncpy(form.exped_form,"            ",12);
   for (i = 0;i < 10;i++)
```

```
{
  strncpy(form.f_etats_poss[i].etats_form,"              ",12);
  form.f_etats_poss[i].dum_et_form = ' ';
  strncpy(form.f_destinataires[i].dest_form,"              ",12);
  form.f_destinataires[i].dum_des_form = ' ';
  strncpy(form.reponse_form[i].nom,"              ",12);
  strncpy(form.reponse_form[i].type,"            ",10);
  form.reponse_form[i].dum_rep = ' ';
 }



/*  appel à la requête et test du status  */

strncpy(trav_bur.program_request_key,"              ",12);
trav_bur.verif_donnees = 'N';
RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FORM_DESC,
                       &form,&trav_bur);
test(RET_STAT);
ccopy(c_trav_bur.c_prog_req_key,
                        trav_bur.program_request_key,12);
while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0") == 0)
{
   trav_bur.verif_donnees = 'Y';
   strncpy(trav_bur.program_request_key,"              ",12);
   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FORM_DESC,
                          &form,&trav_bur);
   test(RET_STAT);
   ccopy(c_trav_bur.c_prog_req_key,
                           trav_bur.program_request_key,12);
}



/*  traduction de la structure TDMS en une structure C  */

ccopy(c_form.c_titre_form,trav_bur.consult_objet,12);
ccopy(c_form.c_type_form,form.type_form,10);
ccopy(c_form.c_exped_form,form.exped_form,12);
```

```
for (i = 0; i < 10; i++)
{
     ccopy(c_form.c_f_etats_poss[i].c_etats_form,
                          form.f_etats_poss[i].etats_form,12);
     c_form.c_f_etats_poss[i].c_dum_et_form
                           = form.f_etats_poss[i].dum_et_form;
     ccopy(c_form.c_f_destinataires[i].c_dest_form,
                        form.f_destinataires[i].dest_form,12);
     c_form.c_f_destinataires[i].c_dum_des_form
                       = form.f_destinataires[i].dum_des_form;
     ccopy(c_form.c_reponse_form[i].c_nom,
                                form.reponse_form[i].nom,12);
     ccopy(c_form.c_reponse_form[i].c_type,
                               form.reponse_form[i].type,10);
     c_form.c_reponse_form[i].c_dum_rep
                              = form.reponse_form[i].dum_rep;
 }


/*  traduction des généralites sur le formulaire  */

/* description de l'objet DSL "ENTITY" lié au formulaire */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"  KEYWORD ARE  \"FORMULAIRE\";\n");
fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"               identifiant_%s,\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"               etat_%s,\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"               echeance,\n");
fprintf(DEFINITIONS,"               statut,\n");
fprintf(DEFINITIONS,"               support;\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  type_%s;\n",
                                      c_form.c_titre_form);
fprintf(DEFINITIONS,"  FORMAT IS \"alphabetique\";\n");
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                                     c_form.c_type_form);
fprintf(DEFINITIONS,"DEFINE ELEMENT  identifiant_%s;\n",
                                     c_form.c_titre_form);
fprintf(DEFINITIONS,"  FORMAT IS \"alphabetique\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                     c_form.c_titre_form);
fprintf(DEFINITIONS,"  FORMAT IS \"alphabetique\";\n");
```

```
   if (c_form.c_f_etats_poss[1].c_etats_form[0] == '\0')
   {
      fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                        c_form.c_f_etats_poss[0].c_etats_form);
   }
   else
   {
      fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                       c_form.c_f_etats_poss[0].c_etats_form);
   }
   nb_et = 1;
   while ((c_form.c_f_etats_poss[nb_et].c_etats_form[0] != '\0')
          && (nb_et < 10))
   {
      if (c_form.c_f_etats_poss[nb_et + 1].c_etats_form[0]
                                                    == '\0')
        {
           fprintf(DEFINITIONS,"                          %s;\n",
                   c_form.c_f_etats_poss[nb_et].c_etats_form);
        }
      else
        {
           fprintf(DEFINITIONS,"                          %s,\n",
                   c_form.c_f_etats_poss[nb_et].c_etats_form);
        }
      nb_et = nb_et + 1;

   }


/* traduction des objets DSL "MESSAGE" liés au formulaire
   du modèle proposé */

   nb_et = 0;
   while ((c_form.c_f_etats_poss[nb_et].c_etats_form[0] != '\0')
          && (nb_et < 10))
   {
      fprintf(DEFINITIONS,"DEFINE MESSAGE  %s_%s_MESS;\n",
                   c_form.c_titre_form,
                   c_form.c_f_etats_poss[nb_et].c_etats_form);
      fprintf(DEFINITIONS,"  KEYWORD ARE  \"FORMULAIRE\";\n");
      fprintf(DEFINITIONS,"  ATTRIBUTE IS  associe_a   \"%_ENT\";\n",
                                          c_form.c_titre_form);
      fprintf(DEFINITIONS,"  CONSISTS OF  type_%s,\n",
                                     c_form.c_titre_form);
      fprintf(DEFINITIONS,"               identifiant_%s,\n",
                                     c_form.c_titre_form);
      fprintf(DEFINITIONS,"               echeance,\n");
      fprintf(DEFINITIONS,"               status,\n");
      fprintf(DEFINITIONS,"               support;\n");
      nb_et = nb_et + 1;
   }
```

```
/*  traduction des réponses attendues  */

nb_rep = 0;
while ((nb_rep < 10)
 && (c_form.c_reponse_form[nb_rep].c_nom[0] != '\0'))
{
   fprintf(DEFINITIONS,"DEFINE RELATION  %s_REP_%s;\n",
     c_form.c_titre_form,c_form.c_reponse_form[nb_rep].c_nom);
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                    c_form.c_titre_form);
   fprintf(DEFINITIONS,"    AS  a_pour_reponse\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                        c_form.c_reponse_form[nb_rep].c_nom);
   fprintf(DEFINITIONS,"    AS  repond_a\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   nb_rep = nb_rep + 1;
}

/*  traduction de l'expéditeur du formulaire  */

fprintf(DEFINITIONS,"DEFINE RELATION  %s_EXP_%s;\n",
                     c_form.c_titre_form,c_form.c_exped_form);
fprintf(DEFINITIONS,"  CONSISTS OF  date_exp_%s;\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_form.c_titre_form);
fprintf(DEFINITIONS,"    AS  expedie_par\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_form.c_exped_form);
fprintf(DEFINITIONS,"    AS  expedie\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  date_exp_%s;\n",
                                    c_form.c_titre_form);
fprintf(DEFINITIONS,"  FORMAT IS  \"99/99/9999\";\n");

/*  traduction des destinataires du formulaire  */

nb_dest = 0;
while ((nb_dest < 10)
 && (c_form.c_f_destinataires[nb_dest].c_dest_form[0] != '\0'))
  {
    fprintf(DEFINITIONS,"DEFINE RELATION  %s_DEST_%s;\n",
            c_form.c_titre_form,
            c_form.c_f_destinataires[nb_dest].c_dest_form);
    fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                    c_form.c_titre_form);
    fprintf(DEFINITIONS,"    AS  destine_a\n");
    fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
    fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
              c_form.c_f_destinataires[nb_dest].c_dest_form);
    fprintf(DEFINITIONS,"    AS  recoit\n");
    fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
    nb_dest = nb_dest + 1;
  }
```

```
   /*  remplissage du formulaire  */

   trav_bur.select_element = 'P';
   while (trav_bur.select_element != 'E')
     {
        if (trav_bur.select_element == 'P')
              choix = 1;
        if (trav_bur.select_element == 'T')
              choix = 2;
        if (trav_bur.select_element == 'V')
              choix = 3;
        switch(choix)
        {
           case 1: add_partie( );
                   break;
           case 2: add_texte( );
                   break;
           case 3: add_variable( );
                   break;
        }
     }


}     /*  fin de add_formulaire  */




/*****************************************************************/
/*****                  FONCTION   ADD_PARTIE                *****/
/*****************************************************************/


add_partie( )
{
   /*  variables  */

   int   i;            /*  compteur  */
   int   nb_emplac;    /*  nombre d'emplacements  */
   int   nb_et;        /*  nombre d'etats  */


   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(part.nom_partie,"           ",12);
   for (i = 0;i <  10;i++)
   {
      part.emplac_partie[i].ligne_part = 0;
      part.emplac_partie[i].colonne_part = 0;
      part.emplac_partie[i].dum_emplac = ' ';
      strncpy(part.p_etats_poss[i].etats_partie,
                          "           ",12);
      part.p_etats_poss[i].dum_et_partie = ' ';
   }
```

```
/*  saisie des informations concernant la partie  */


strncpy(trav_bur.program_request_key,"            ",12);
trav_bur.verif_donnees = 'N';
RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PART_DESC,
                       &part,&trav_bur);
test(RET_STAT);
ccopy(c_trav_bur.c_prog_req_key,
                  trav_bur.program_request_key,12);
while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION/0")
                                                    == 0)
  {
    trav_bur.verif_donnees = 'Y';
    strncpy(trav_bur.program_request_key,"            ",12);
    RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PART_DESC,
                           &part,&trav_bur);
    test(RET_STAT);
    ccopy(c_trav_bur.c_prog_req_key,
                        trav_bur.program_request_key,12);
   }



/*  traduction de la structure TDMS en une structure C  */

ccopy(c_part.c_nom_partie,part.nom_partie,12);
for (i = 0;i < 10;i++)
{
  c_part.c_emplac_partie[i].c_ligne_part
                          = part.emplac_partie[i].ligne_part;
  c_part.c_emplac_partie[i].c_colonne_part
                         = part.emplac_partie[i].colonne_part;
  c_part.c_emplac_partie[i].c_dum_emplac
                          = part.emplac_partie[i].dum_emplac;
  ccopy(c_part.c_p_etats_poss[i].c_etats_partie,
                        part.p_etats_poss[i].etats_partie,12);
  c_part.c_p_etats_poss[i].c_dum_et_partie
                         = part.p_etats_poss[i].dum_et_partie;
}
```

```
/*  description de l'objet DSL "ENTITY" lié à la partie  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                        c_part.c_nom_partie);
fprintf(DEFINITIONS,"  ATTRIBUTE IS  PARTIE_DE  \"%s_ENT\";\n",
                                        c_form.c_titre_form);
fprintf(DEFINITIONS,"  CONSISTS OF  etat_%s;\n",
                                        c_part.c_nom_partie);
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                        c_part.c_nom_partie);
fprintf(DEFINITIONS,"  FORMAT IS \"alphabetique\";\n");
if (c_part.c_p_etats_poss[1].c_etats_partie[0] == '\0')
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                    c_part.c_p_etats_poss[0].c_etats_partie);
}
else
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                   c_part.c_p_etats_poss[0].c_etats_partie);
}
nb_et = 1;
while ((c_part.c_p_etats_poss[nb_et].c_etats_partie[0] != '\0')
       && (nb_et < 10))
{
 if (c_part.c_p_etats_poss[nb_et + 1].c_etats_partie[0] == '\0')
     {
        fprintf(DEFINITIONS,"                              %s;\n",
             c_part.c_p_etats_poss[nb_et].c_etats_partie);
     }
   else
     {
        fprintf(DEFINITIONS,"                              %s,\n",
             c_part.c_p_etats_poss[nb_et].c_etats_partie);
     }
   nb_et = nb_et + 1;

}


/*  description de la relation traitant le lien avec
    le formulaire  */

fprintf(DEFINITIONS,"DEFINE RELATION  %s_COMP_%s;\n",
                   c_form.c_titre_form,c_part.c_nom_partie);
fprintf(DEFINITIONS,"  CONSISTS OF  emplacement_%s;\n",
                                        c_part.c_nom_partie);
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_form.c_titre_form);
fprintf(DEFINITIONS,"    AS  se_decompose_en\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"1_N\";\n");
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_part.c_nom_partie);
fprintf(DEFINITIONS,"    AS  fait_partie_de\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"1_N\";\n");
```

```
   fprintf(DEFINITIONS,"DEFINE ELEMENT emplacement_%s;\n",
                                          c_part.c_nom_partie);
   if ((c_part.c_emplac_partie[1].c_ligne_part != 0)
       && (c_part.c_emplac_partie[1].c_colonne_part != 0))
     {
        fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\",\n",
                     c_part.c_emplac_partie[0].c_ligne_part,
                  c_part.c_emplac_partie[0].c_colonne_part);
     }
   else
    {
        fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\";\n",
                     c_part.c_emplac_partie[0].c_ligne_part,
                  c_part.c_emplac_partie[0].c_colonne_part);
     }
   nb_emplac = 1;
   while ((c_part.c_emplac_partie[nb_emplac].c_ligne_part != 0)
     && (c_part.c_emplac_partie[nb_emplac].c_colonne_part != 0)
     && (nb_emplac < 10))
   {
     if ((c_part.c_emplac_partie[nb_emplac + 1].c_ligne_part != 0)
     && (c_part.c_emplac_partie[nb_emplac + 1].c_colonne_part != 0))
        {
          fprintf(DEFINITIONS,"                          \"%d_%d\",\n",
                c_part.c_emplac_partie[nb_emplac].c_ligne_part,
             c_part.c_emplac_partie[nb_emplac].c_colonne_part);
        }
      else
       {
         fprintf(DEFINITIONS,"                          \"%d_%d\";\n",
               c_part.c_emplac_partie[nb_emplac].c_ligne_part,
            c_part.c_emplac_partie[nb_emplac].c_colonne_part);
        }
       nb_emplac = nb_emplac + 1;
    }

}
```

```
/*****************************************************************/
/*****                FONCTION  ADD_TEXTE                    *****/
/*****************************************************************/



add_texte( )
{

   /*  variables  */

   int  nb_emplac;
   int  i;

   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(elt_txt.nom_texte,"            ",12);
   strncpy(elt_txt.format,"                                ",30);
   strncpy(elt_txt.contenu,"                                ",30);
   for (i = 0;i <  10;i++)
   {
      elt_txt.emplac_texte[i].ligne_txt = 0;
      elt_txt.emplac_texte[i].colonne_txt = 0;
      elt_txt.emplac_texte[i].dum_txt = ' ';
   }



   /*  saisie des informations concernant l'élément
       de texte  */


   strncpy(trav_bur.program_request_key,"            ",12);
   trav_bur.verif_donnees = 'N';
   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&TEXTE_DESC,
                          &elt_txt,&trav_bur);
   test(RET_STAT);
   ccopy(c_trav_bur.c_prog_req_key,
                          trav_bur.program_request_key,12);
   while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0") == 0)
     {
       trav_bur.verif_donnees = 'Y';
       strncpy(trav_bur.program_request_key,"            ",12);
       RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&TEXTE_DESC,
                              &elt_txt,&trav_bur);
/*       test(RET_STAT);    */
       ccopy(c_trav_bur.c_prog_req_key,
                              trav_bur.program_request_key,12);
      }
```

```
/*  traduction de la structure TDMS en une structure C  */

ccopy(c_elt_txt.c_nom_texte,elt_txt.nom_texte,12);
ccopy(c_elt_txt.c_format,elt_txt.format,30);
ccopy(c_elt_txt.c_contenu,elt_txt.contenu,30);
for (i = 0;i <  10;i++)
{
   c_elt_txt.c_emplac_texte[i].c_ligne_txt
                          = elt_txt.emplac_texte[i].ligne_txt;
   c_elt_txt.c_emplac_texte[i].c_colonne_txt
                        = elt_txt.emplac_texte[i].colonne_txt;
   c_elt_txt.c_emplac_texte[i].c_dum_txt
                           = elt_txt.emplac_texte[i].dum_txt;
}



/*  description de l'objet DSL "ENTITY" lie a l'élément de texte  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"  ATTRIBUTE IS  ELT_TXT_DE  \"%s_ENT\";\n",
                                      c_part.c_nom_partie);
fprintf(DEFINITIONS,"  CONSISTS OF  format_%s,\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"                contenu_%s;\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"DEFINE ELEMENT  format_%s;\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%s\";\n",
                                     c_elt_txt.c_format);
fprintf(DEFINITIONS,"DEFINE ELEMENT  contenu_%s;\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%s\";\n",
                                      c_elt_txt.c_contenu);
```

```
/*  description de la relation traitant le lien avec
    la partie  */

fprintf(DEFINITIONS,"DEFINE RELATION  %s_COMT_%s;\n",
                    c_part.c_nom_partie,c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"  CONSISTS OF  emplacement_%s;\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                     c_part.c_nom_partie);
fprintf(DEFINITIONS,"    AS  se_decompose_en\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                    c_elt_txt.c_nom_texte);
fprintf(DEFINITIONS,"    AS  fait_partie_de\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"1_N\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  emplacement_%s;\n",
                                   c_elt_txt.c_nom_texte);
if ((c_elt_txt.c_emplac_texte[1].c_ligne_txt != 0)
    && (c_elt_txt.c_emplac_texte[1].c_colonne_txt != 0))
  {
    fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\",\n",
                c_elt_txt.c_emplac_texte[0].c_ligne_txt,
              c_elt_txt.c_emplac_texte[0].c_colonne_txt);
  }
else
  {
    fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\";\n",
                c_elt_txt.c_emplac_texte[0].c_ligne_txt,
              c_elt_txt.c_emplac_texte[0].c_colonne_txt);
  }
 nb_emplac = 1;
 while ((c_elt_txt.c_emplac_texte[nb_emplac].c_ligne_txt != 0)
    && (c_elt_txt.c_emplac_texte[nb_emplac].c_colonne_txt !=0)
    && (nb_emplac < 10))
  {
    if ((c_elt_txt.c_emplac_texte[nb_emplac + 1].c_ligne_txt != 0)
    && (c_elt_txt.c_emplac_texte[nb_emplac + 1].c_colonne_txt != 0))
    {
      fprintf(DEFINITIONS,"                              \"%d_%d\",\n",
                c_elt_txt.c_emplac_texte[nb_emplac].c_ligne_txt,
              c_elt_txt.c_emplac_texte[nb_emplac].c_colonne_txt);
    }
    else
    {
      fprintf(DEFINITIONS,"                              \"%d_%d\";\n",
                c_elt_txt.c_emplac_texte[nb_emplac].c_ligne_txt,
              c_elt_txt.c_emplac_texte[nb_emplac].c_colonne_txt);
      }
      nb_emplac = nb_emplac + 1;

  }

}
```

```
/*****************************************************************/
/*****          FONCTION  ADD_VARIABLE                      *****/
/*****************************************************************/


add_variable( )
{
   /*  variables  */

   int   i;            /*  compteur  */
   int   nb_emplac;    /*  nombre d'emplacements  */


   /*  initialisation de la structure qui contiendra
       les informations  */

   strncpy(elt_var.nom_var,"            ",12);
   strncpy(elt_var.format,"                                ",30);
   strncpy(elt_var.val_def,"                                ",30);
   strncpy(elt_var.txt_cor,"            ",12);
   for (i = 0;i <  10;i++)
   {
      elt_var.emplac_var[i].ligne_var = 0;
      elt_var.emplac_var[i].colonne_var = 0;
      elt_var.emplac_var[i].dum_var = ' ';
   }



   /*  saisie des informations concernant l'élément variable  */


   strncpy(trav_bur.program_request_key,"            ",12);
   trav_bur.verif_donnees = 'N';
   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&VAR_DESC,
                          &elt_var,&trav_bur);
   test(RET_STAT);
   ccopy(c_trav_bur.c_prog_req_key,
                       trav_bur.program_request_key,12);
   while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0") == 0)
     {
       trav_bur.verif_donnees = 'Y';
       strncpy(trav_bur.program_request_key,"            ",12);
       RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&VAR_DESC,
                              &elt_var,&trav_bur);
       test(RET_STAT);
       ccopy(c_trav_bur.c_prog_req_key,
                              trav_bur.program_request_key,12);
     }
```

```
/*  traduction de la structure TDMS en une structure C  */

ccopy(c_elt_var.c_nom_var,elt_var.nom_var,12);
ccopy(c_elt_var.c_format,elt_var.format,30);
ccopy(c_elt_var.c_val_def,elt_var.val_def,30);
ccopy(c_elt_var.c_txt_cor,elt_var.txt_cor,12);
for (i = 0;i <  10;i++)
{
   c_elt_var.c_emplac_var[i].c_ligne_var
                        = elt_var.emplac_var[i].ligne_var;
   c_elt_var.c_emplac_var[i].c_colonne_var
                       = elt_var.emplac_var[i].colonne_var;
   c_elt_var.c_emplac_var[i].c_dum_var
                         = elt_var.emplac_var[i].dum_var;
}


/*  description de l'objet DSL "ENTITY" lié à
    l'élément variable  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"  ATTRIBUTE IS  ELT_VAR_DE  \"%s_ENT\";\n",
                                    c_part.c_nom_partie);
fprintf(DEFINITIONS,"  CONSISTS OF  format_%s,\n",
                                    c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"                val_def_%s;\n",
                                    c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"DEFINE ELEMENT  format_%s;\n",
                                    c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%s\";\n",
                                     c_elt_var.c_format);
fprintf(DEFINITIONS,"DEFINE ELEMENT  val_def_%s;\n",
                                    c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%s\";\n",
                                    c_elt_var.c_val_def);


/*  description de la relation traitant le lien avec
    le texte  */

fprintf(DEFINITIONS,"DEFINE RELATION %s_CORR_%s;\n",
                    c_elt_var.c_nom_var,c_elt_var.c_txt_cor);
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_elt_var.c_nom_var);
fprintf(DEFINITIONS,"    AS  correspond_a\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_elt_var.c_txt_cor);
fprintf(DEFINITIONS,"    AS  correspond_a\n");
fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_1\";\n");
```

```
   /*  description de la relation traitant le lien avec
       la partie  */

   fprintf(DEFINITIONS,"DEFINE RELATION  %s_COMV_%s;\n",
                        c_part.c_nom_partie,c_elt_var.c_nom_var);
   fprintf(DEFINITIONS,"  CONSISTS OF  emplacement_%s;\n",
                                        c_elt_var.c_nom_var);
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_part.c_nom_partie);
   fprintf(DEFINITIONS,"    AS  se_decompose_en\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
   fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_elt_var.c_nom_var);
   fprintf(DEFINITIONS,"    AS  fait_partie_de\n");
   fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"1_N\";\n");
   fprintf(DEFINITIONS,"DEFINE ELEMENT emplacement_%s;\n",
                                       c_elt_var.c_nom_var);
   if ((c_elt_var.c_emplac_var[1].c_ligne_var != 0)
       && (c_elt_var.c_emplac_var[1].c_colonne_var != 0))
     {
       fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\",\n",
                     c_elt_var.c_emplac_var[0].c_ligne_var,
                  c_elt_var.c_emplac_var[0].c_colonne_var);
     }
   else
     {
       fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  \"%d_%d\";\n",
                     c_elt_var.c_emplac_var[0].c_ligne_var,
                  c_elt_var.c_emplac_var[0].c_colonne_var);
     }
   nb_emplac = 1;
   while ((c_elt_var.c_emplac_var[nb_emplac].c_ligne_var != 0)
      && (c_elt_var.c_emplac_var[nb_emplac].c_colonne_var != 0)
      && (nb_emplac < 10))
   {
    if ((c_elt_var.c_emplac_var[nb_emplac + 1].c_ligne_var != 0)
    && (c_elt_var.c_emplac_var[nb_emplac + 1].c_colonne_var != 0))
     {
        fprintf(DEFINITIONS,"                              \"%d_%d\",\n",
                   c_elt_var.c_emplac_var[nb_emplac].c_ligne_var,
              c_elt_var.c_emplac_var[nb_emplac].c_colonne_var);
     }
   else
    {
        fprintf(DEFINITIONS,"                              \"%d_%d\";\n",
                   c_elt_var.c_emplac_var[nb_emplac].c_ligne_var,
              c_elt_var.c_emplac_var[nb_emplac].c_colonne_var);
     }
    nb_emplac = nb_emplac + 1;
 }

}
```

```
/*****************************************************************/
/*****              FONCTION ADD_DOSSIER                     *****/
/*****************************************************************/

add_dossier( )
{
   /*  déclaration des variables internes  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  compteur: nombre d'etats  */
   int   nb_dest;    /*  compteur: nombre de destinataires  */
   int   nb_cons;    /*  compteur: nombre de constituants  */

   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(doss.nom_dossier,trav_bur.consult_objet,12);
   strncpy(doss.exped_doss,"            ",12);
   for (i = 0; i < 10; i++)
   {
     strncpy(doss.do_etats_poss[i].etats_doss,"            ",12);
     doss.do_etats_poss[i].dum_et_doss = ' ';
     strncpy(doss.do_destinataires[i].dest_doss,"            ",
                                                          12);
     doss.do_destinataires[i].dum_des_doss = ' ';
   }
   for (i = 0; i < 20; i++)
   {
     strncpy(doss.constitue_doss[i].const_doss,"            ",
                                                         12);
     strncpy(doss.constitue_doss[i].type_const_doss,
                                              "          ",10);
     doss.constitue_doss[i].dum_const = ' ';
   }

    /*  appel de la requête d'affichage du dossier  */

    strncpy(trav_bur.program_request_key,"            ",12);
    trav_bur.verif_donnees = 'N';
    RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOSS_DESC,
                           &doss,&trav_bur);
/*    test(RET_STAT);  */
    ccopy(c_trav_bur.c_prog_req_key,
                              trav_bur.program_request_key,12);
    while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0") == 0)
      {
        trav_bur.verif_donnees = 'Y';
        strncpy(trav_bur.program_request_key,"            ",12);
        RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOSS_DESC,
                               &doss,&trav_bur);
/*        test(RET_STAT);   */
        ccopy(c_trav_bur.c_prog_req_key,
                           trav_bur.program_request_key,12);
      }
```

```
/*  traduction de la structure TDMS en une structure C  */

ccopy(c_doss.c_nom_dossier,trav_bur.consult_objet,12);
ccopy(c_doss.c_exped_doss,doss.exped_doss,12);
for (i = 0; i < 10; i++)
  {
     ccopy(c_doss.c_do_etats_poss[i].c_etats_doss,
                        doss.do_etats_poss[i].etats_doss,12);
     c_doss.c_do_etats_poss[i].c_dum_et_doss
                         = doss.do_etats_poss[i].dum_et_doss;
     ccopy(c_doss.c_do_destinataires[i].c_dest_doss,
                       doss.do_destinataires[i].dest_doss,12);
     c_doss.c_do_destinataires[i].c_dum_des_doss
                      = doss.do_destinataires[i].dum_des_doss;
  }
for (i = 0; i < 20; i++)
  {
     ccopy(c_doss.c_constitue_doss[i].c_const_doss,
                        doss.constitue_doss[i].const_doss,12);
     ccopy(c_doss.c_constitue_doss[i].c_type_const_doss,
                    doss.constitue_doss[i].type_const_doss,10);
     c_doss.c_constitue_doss[i].c_dum_const
                         = doss.constitue_doss[i].dum_const;
  }



/*  L'information voulue est maintenant dans la structure DOSS.  */
/*  Je traduis en DSL.  */


/*  Description de l'objet DSL "ENTITY" lié au dossier.  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_doss.c_nom_dossier);
fprintf(DEFINITIONS,"  KEYWORD ARE  \"DOSSIER\";\n");
fprintf(DEFINITIONS,"  CONSISTS OF  identifiant_%s,\n",
                                    c_doss.c_nom_dossier);
fprintf(DEFINITIONS,"               etat_%s,\n",
                                    c_doss.c_nom_dossier);
fprintf(DEFINITIONS,"               echeance,\n");
fprintf(DEFINITIONS,"               support;\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  identifiant_%s;\n",
                                   c_doss.c_nom_dossier);
fprintf(DEFINITIONS,"  FORMAT IS  \"alphabetique\";\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                    c_doss.c_nom_dossier);
```

```
if (c_doss.c_do_etats_poss[1].c_etats_doss[0] == '\0')
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                       c_doss.c_do_etats_poss[0].c_etats_doss);
}
else
{
  fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                       c_doss.c_do_etats_poss[0].c_etats_doss);
}

nb_et = 1;
while ((c_doss.c_do_etats_poss[nb_et].c_etats_doss[0] != '\0')
       && (nb_et <10))
{
 if (c_doss.c_do_etats_poss[nb_et + 1].c_etats_doss[0] == '\0')
     {
        fprintf(DEFINITIONS,"                              %s;\n",
                  c_doss.c_do_etats_poss[nb_et].c_etats_doss);

     }
   else
     {
        fprintf(DEFINITIONS,"                              %s,\n",
                 c_doss.c_do_etats_poss[nb_et].c_etats_doss);
     }
   nb_et = nb_et + 1;

}

/* traduction des objets DSL "MESSAGE" liés au dossier
   du modèle proposé */

nb_et = 0;
while ((c_doss.c_do_etats_poss[nb_et].c_etats_doss[0] != '\0')
       && (nb_et < 10))
{
   fprintf(DEFINITIONS,"DEFINE MESSAGE  %s_%s_MESS;\n",
             c_doss.c_nom_dossier,
             c_doss.c_do_etats_poss[nb_et].c_etats_doss);
   fprintf(DEFINITIONS,"  KEYWORD ARE  \"DOSSIER\";\n");
   fprintf(DEFINITIONS,"  ATTRIBUTE IS  associe_a   \"%s_ENT\";\n",
                                 c_doss.c_nom_dossier);
   fprintf(DEFINITIONS,"  CONSISTS OF  identifiant_%s,\n",
                                 c_doss.c_nom_dossier);
   fprintf(DEFINITIONS,"                  echeance,\n");
   fprintf(DEFINITIONS,"                  support;\n");
   nb_et = nb_et + 1;
}
```

```
/*  traduction de l'expéditeur du dossier  */

if (c_doss.c_exped_doss[0] != '\0')
{
  fprintf(DEFINITIONS,"DEFINE RELATION  %s_EXP_%s;\n",
                  c_doss.c_nom_dossier,c_doss.c_exped_doss);
  fprintf(DEFINITIONS,"  CONSISTS OF  date_exp_%s;\n",
                                  c_doss.c_nom_dossier);
  fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                  c_doss.c_nom_dossier);
  fprintf(DEFINITIONS,"    AS  expedie_par\n");
  fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
  fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                  c_doss.c_exped_doss);
  fprintf(DEFINITIONS,"    AS  expedie\n");
  fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
  fprintf(DEFINITIONS,"DEFINE ELEMENT  date_exp_%s;\n",
                                 c_doss.c_nom_dossier);
  fprintf(DEFINITIONS,"  FORMAT IS  \"99/99/9999\";\n");
}


/*  traduction des destinataires du dossier  */

nb_dest = 0;
while ((nb_dest < 10)
 && (c_doss.c_do_destinataires[nb_dest].c_dest_doss[0] != '\0'))
  {
    fprintf(DEFINITIONS,"DEFINE RELATION  %s_DEST_%s;\n",
            c_doss.c_nom_dossier,
            c_doss.c_do_destinataires[nb_dest].c_dest_doss);
    fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                  c_doss.c_nom_dossier);
    fprintf(DEFINITIONS,"    AS  destine_a\n");
    fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
    fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
             c_doss.c_do_destinataires[nb_dest].c_dest_doss);
    fprintf(DEFINITIONS,"    AS  recoit\n");
    fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
    nb_dest = nb_dest + 1;
  }
```

```
   /*  traduction des constituants du dossier  */

   nb_cons = 0;
   while ((nb_cons < 20)
    && (c_doss.c_constitue_doss[nb_cons].c_const_doss[0] != '\0'))
   {
      fprintf(DEFINITIONS,"DEFINE RELATION  %s_CONS_%s;\n",
              c_doss.c_nom_dossier,
              c_doss.c_constitue_doss[nb_cons].c_const_doss);
      fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                                    c_doss.c_nom_dossier);
      fprintf(DEFINITIONS,"    AS  contient\n");
      fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
      fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                c_doss.c_constitue_doss[nb_cons].c_const_doss);
      fprintf(DEFINITIONS,"    AS  est_contenu_dans\n");
      fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_1\";\n");
      nb_cons = nb_cons + 1;
   }

}

/**************  fin de la traduction du dossier          ******************/
```

```
/*****************************************************************/
/*****                  FONCTION  ADD_FICHIER                *****/
/*****************************************************************/


add_fichier( )
{
   /*  declaration des variables internes  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  compteur: nombre d'etats  */

   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(fich.nom_fichier,trav_bur.consult_objet,12);
   strncpy(fich.constitue_fich.const_fich,"            ",12);
   strncpy(fich.constitue_fich.type_const_fich,"          ",10);
   for (i = 0; i < 10; i++)
     {
        strncpy(fich.fi_etats_poss[i].etats_fich,
                                              "            ",12);
        fich.fi_etats_poss[i].dum_et_fich = ' ';
     }


    /*  appel de la requête d'affichage du fichier  */

    strncpy(trav_bur.program_request_key,"            ",12);
    trav_bur.verif_donnees = 'N';
    RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FICH_DESC,
                           &fich,&trav_bur);
/*    test(RET_STAT);     */
    ccopy(c_trav_bur.c_prog_req_key,
                            trav_bur.program_request_key,12);
    while (strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0") == 0)
      {
        trav_bur.verif_donnees = 'Y';
        strncpy(trav_bur.program_request_key,"            ",12);
        RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FICH_DESC,
                                                &fich,&trav_bur);
/*        test(RET_STAT);     */
        ccopy(c_trav_bur.c_prog_req_key,
                               trav_bur.program_request_key,12);
      }
```

```
/*  traduction de la structure TDMS en une structure C  */

ccopy(c_fich.c_nom_fichier,trav_bur.consult_objet,12);
ccopy(c_fich.c_constitue_fich.c_const_fich,
                    fich.constitue_fich.const_fich,12);
ccopy(c_fich.c_constitue_fich.c_type_const_fich,
                fich.constitue_fich.type_const_fich,10);
for (i = 0; i < 10; i++)
  {
     ccopy(c_fich.c_fi_etats_poss[i].c_etats_fich,
                  fich.fi_etats_poss[i].etats_fich,12);
     c_fich.c_fi_etats_poss[i].c_dum_et_fich
                  = fich.fi_etats_poss[i].dum_et_fich;
   }


/*  L'information voulue est maintenant dans la structure FICH.  */
/*  Je traduis en DSL.  */

/*  Description de l'objet DSL "ENTITY" lié au fichier.  */

fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                    c_fich.c_nom_fichier);
fprintf(DEFINITIONS,"  KEYWORD ARE  \"FICHIER\";\n");
fprintf(DEFINITIONS,"  CONSISTS OF  etat_%s,\n",
                                    c_fich.c_nom_fichier);
fprintf(DEFINITIONS,"               support;\n");
fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                   c_fich.c_nom_fichier);
if (c_fich.c_fi_etats_poss[1].c_etats_fich[0] == '\0')
{
    fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                     c_fich.c_fi_etats_poss[0].c_etats_fich);
}
else
{
    fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                    c_fich.c_fi_etats_poss[0].c_etats_fich);
}
nb_et = 1;
while ((c_fich.c_fi_etats_poss[nb_et].c_etats_fich[0] != '\0')
       && (nb_et <10))
{
   if (c_fich.c_fi_etats_poss[nb_et + 1].c_etats_fich[0] == '\0')
     {
        fprintf(DEFINITIONS,"                              %s;\n",
                c_fich.c_fi_etats_poss[nb_et].c_etats_fich);
     }
   else
     {
        fprintf(DEFINITIONS,"                              %s,\n",
                c_fich.c_fi_etats_poss[nb_et].c_etats_fich);
     }
   nb_et = nb_et + 1;

}
```

```
   /*  traduction du constituant du fichier  */

  fprintf(DEFINITIONS,"DEFINE RELATION  %s_CONS_%s;\n",
         c_fich.c_nom_fichier,
         c_fich.c_constitue_fich.c_const_fich);
  fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",c_fich.c_nom_fichier);
  fprintf(DEFINITIONS,"    AS  contient\n");
  fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_N\";\n");
  fprintf(DEFINITIONS,"  RELATES  %s_ENT\n",
                   c_fich.c_constitue_fich.c_const_fich);
  fprintf(DEFINITIONS,"    AS  est_contenu_dans\n");
  fprintf(DEFINITIONS,"    WITH CONNECTIVITY  \"0_1\";\n");


}

/**************  fin de la traduction du fichier          ******************/




/*****************************************************************/
/*****                  FONCTION  ADD_PILE                   *****/
/*****************************************************************/


add_pile( )
{
   /*  déclaration des variables internes  */

   int   i;          /*  compteur  */
   int   nb_et;      /*  compteur: nombre d'etats  */

   /*  initialisation de la structure qui contiendra les
       informations  */

   strncpy(tas.nom_pile,trav_bur.consult_objet,12);
   for (i = 0; i < 10; i++)
   {
     strncpy(tas.pi_etats_poss[i].etats_pile,"           ",12);
     tas.pi_etats_poss[i].dum_et_pile = ' ';
   }
```

```
/*  appel de la requête d'affichage du fichier  */

strncpy(trav_bur.program_request_key,"            ",12);
trav_bur.verif_donnees = 'N';
RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PILE_DESC,
                       &tas,&trav_bur);
test(RET_STAT);
ccopy(c_trav_bur.c_prog_req_key,
                       trav_bur.program_request_key,12);
while(strcmp(c_trav_bur.c_prog_req_key,"VERIFICATION\0")
                                                  == 0)
{
   trav_bur.verif_donnees = 'Y';
   strncpy(trav_bur.program_request_key,"            ",12);
   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PILE_DESC,
                          &tas,&trav_bur);
   test(RET_STAT);
   ccopy(c_trav_bur.c_prog_req_key,
                       trav_bur.program_request_key,12);
}


/*  traduction de la structure de TDMS en C  */


ccopy(c_tas.c_nom_pile,trav_bur.consult_objet,12);
for (i = 0; i < 10; i++)
{
   ccopy(c_tas.c_pi_etats_poss[i].c_etats_pile,
            tas.pi_etats_poss[i].etats_pile,12);
   c_tas.c_pi_etats_poss[i].c_dum_et_pile
            = tas.pi_etats_poss[i].dum_et_pile;
}
```

```
   /*   L'information voulue est maintenant dans la structure TAS.   */
   /*   Je traduis en DSL  */


   fprintf(DEFINITIONS,"DEFINE ENTITY  %s_ENT;\n",
                                       c_tas.c_nom_pile);
   fprintf(DEFINITIONS,"  KEYWORD ARE  \"PILE\";\n");
   fprintf(DEFINITIONS,"  CONSISTS OF  etat_%s,\n",
                                       c_tas.c_nom_pile);
   fprintf(DEFINITIONS,"               support;\n");
   fprintf(DEFINITIONS,"DEFINE ELEMENT  etat_%s;\n",
                                     c_tas.c_nom_pile);
   if (c_tas.c_pi_etats_poss[1].c_etats_pile[0] == '\0')
   {
      fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s;\n",
                 c_tas.c_pi_etats_poss[0].c_etats_pile);
   }
   else
   {
      fprintf(DEFINITIONS,"  DOMAIN OF VALUE ARE  %s,\n",
                 c_tas.c_pi_etats_poss[0].c_etats_pile);
   }
   nb_et = 1;
   while ((c_tas.c_pi_etats_poss[nb_et].c_etats_pile[0] != '\0')
          && (nb_et <10))
   {
    if (c_tas.c_pi_etats_poss[nb_et + 1].c_etats_pile[0] == '\0')
        {
           fprintf(DEFINITIONS,"                              %s;\n",
                   c_tas.c_pi_etats_poss[nb_et].c_etats_pile);
        }
      else
        {
           fprintf(DEFINITIONS,"                              %s ,\n",
                    c_tas.c_pi_etats_poss[nb_et].c_etats_pile);
        }
      nb_et = nb_et + 1;

   }


}

/**************  fin de la traduction de la pile          ******************/
```

2.3.3 LA MISE DANS LA BASE DE DONNEES DSL

A l'issue de l'exécution du programme "BUREAU", on dispose de la description des objets informationnels du bureau, sous forme de phrases DSL, dans le fichier "DEFINITIONS" du programme, qui correspond au fichier "DEFBUREAU.DSL".

Pour terminer la transformation des objets de bureau, en DSL, il faut encore entrer les phrases DSL dans une base de données des spécifications DSL.

Pour ce, on doit tout d'abord initialiser une base de données des spécifications. Cela se fait au moyen de la commande:

"INITDB nom de la base de données".

On appellera la base de données "DATABASE".

Ensuite, on fait appel au langage des commandes de DSL-SPEC:

- La commande "SET DATABASE" permet de spécifier le nom du fichier qui contient la base de données des spécifications de l'utilisateur:

  "SET DB=DATABASE.DBF OUTPUT=RAPPORT.PTR".

- La commande "IP" permet d'ajouter dans la base de données les informations spécifiées à l'aide de phrases DSL dans le fichier "DEFBUREAU.DSL". Un compte-rendu des spécifications est généré sous forme de rapport dans le fichier "RAPPORT.PTR" (voir "Les rapports documentaires"). Le rapport fournit la liste des phrases DSL données en entrée de la commande dans l'ordre où elles apparaissaient, et pour chacune d'elles, précise si la création a pu se réaliser et sinon pourquoi.

  La commande s'énonce comme suit:

  "IP INPUT=DEFBUREAU.DSL".

2.3.4 LES RAPPORTS DOCUMENTAIRES

La base de données des spécifications en DSL peut être exploitée pour la production de rapports documentaires.
Voyons quelques exemples de rapports.

A. Les rapports fournis par l'exécution de la commande "IP"

A.1. Rapport d'ajout d'un message.

```
Interactive Design Approach    IDA2.0RO                    Page 1
                               FNDP - Namur VAX/VMS

                        Input Processor Source Listing

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=MESSAGE.DSL
   SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

    1 DEFINE  ELEMENT  contenu;
    2    FORMAT IS  "texte";
    3 DEFINE  ELEMENT  echeance;
    4    FORMAT IS  "99/99/99";
    5 DEFINE  ELEMENT  statut;
    6    DOMAIN OF VALUE ARE  original,copie;
    7 DEFINE  ELEMENT  support;
    8    FORMAT IS  "alphabetique";
    9 DEFINE  GROUP sujet;
   10    CONSISTS OF  5 mot_cle;
   11 DEFINE  ELEMENT  mot_cle;
   12    FORMAT IS  "alphabetique";
   13 DEFINE  ELEMENT  corps;
   14    FORMAT IS  "texte";
   15 DEFINE  ELEMENT  paragraphes_statiques;
   16    FORMAT IS  "booleen";
   17 DEFINE ENTITY  tel_contact_ENT;
   18   KEYWORD ARE  "MSGE";
   19   CONSISTS OF  type_tel_contact,
   20                explication_tel_contact,
   21                contenu,
   22                etat_tel_contact,
   23                echeance,
   24                statut,
   25                support;
   26 DEFINE ELEMENT  type_tel_contact;
   27   DOMAIN OF VALUE ARE  telephone;
```

```
28 DEFINE ELEMENT  explication_tel_contact;
29 DEFINE ELEMENT  etat_tel_contact;
30   DOMAIN OF VALUE ARE  envoye_secr,
31                        traite_t;
32 DEFINE MESSAGE  tel_contact_envoye_secr_MESS;
33   KEYWORD ARE  "MSGE";
34   ATTRIBUTE IS  associe_a   "tel_contact_ENT";
35   CONSISTS OF  type_tel_contact,
36                explication_tel_contact,
37                contenu,
38                echeance,
39                statut,
40                support;
41 DEFINE MESSAGE  tel_contact_traite_t_MESS;
42   KEYWORD ARE  "MSGE";
43   ATTRIBUTE IS  associe_a   "tel_contact_ENT";
44   CONSISTS OF  type_tel_contact,
45                explication_tel_contact,
46                contenu,
47                echeance,
48                statut,
49                support;
50 DEFINE RELATION  tel_contact_REP_lettre_accep;
51   RELATES  tel_contact_ENT
52     AS  a_pour_reponse
53     WITH CONNECTIVITY  "0_N";
54   RELATES  lettre_accep_ENT
55     AS  repond_a
56     WITH CONNECTIVITY  "0_N";
57 DEFINE RELATION  tel_contact_REP_dde_inscr;
58   RELATES  tel_contact_ENT
59     AS  a_pour_reponse
60     WITH CONNECTIVITY  "0_N";
61   RELATES  dde_inscr_ENT
62     AS  repond_a
63     WITH CONNECTIVITY  "0_N";
64 DEFINE RELATION  tel_contact_EXP_etudiant;
65   CONSISTS OF  date_exp_tel_contact;
66   RELATES  tel_contact_ENT
67     AS  expedie_par
68     WITH CONNECTIVITY  "0_N";
69   RELATES  etudiant_ENT
70     AS  expedie
71     WITH CONNECTIVITY  "0_N";
72 DEFINE ELEMENT  date_exp_tel_contact;
73   FORMAT IS  "99/99/9999";
74 DEFINE RELATION  tel_contact_DEST_secret_adm;
75  RELATES  tel_contact_ENT
76  AS destine_a
77  WITH CONNECTIVITY "0_N";
78  RELATES  secret_adm_ENT
79  AS recoit
80  WITH CONNECTIVITY "0_N";
```

A.2. Rapport d'ajout d'un document.

```
Interactive Design Approach    IDA2.0R0                    Page 1
                               FNDP - Namur VAX/VMS

                          Input Processor Source Listing

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=DOCUMENT.DSL
  SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

   1 DEFINE  ELEMENT  contenu;
   2    FORMAT IS  "texte";
   3 DEFINE  ELEMENT  echeance;
   4    FORMAT IS  "99/99/99";
   5 DEFINE  ELEMENT  statut;
   6    DOMAIN OF VALUE ARE  original,copie;
   7 DEFINE  ELEMENT  support;
   8    FORMAT IS  "alphabetique";
   9 DEFINE  GROUP sujet;
  10    CONSISTS OF  5 mot_cle;
  11 DEFINE  ELEMENT  mot_cle;
  12    FORMAT IS  "alphabetique";
  13 DEFINE  ELEMENT  corps;
  14    FORMAT IS  "texte";
  15 DEFINE  ELEMENT  paragraphes_statiques;
  16    FORMAT IS  "booleen";
  17 DEFINE ENTITY  lettre_cont_ENT;
  18   KEYWORD ARE  "DOCUMENT";
  19   CONSISTS OF  type_lettre_cont,
  20                explication_lettre_cont,
  21                contenu,
  22                etat_lettre_cont,
  23                echeance,
  24                statut,
  25                support;
  26 DEFINE ELEMENT  type_lettre_cont;
  27   DOMAIN OF VALUE ARE  lettre;
  28 DEFINE ELEMENT  explication_lettre_cont;
  29 DEFINE ELEMENT  etat_lettre_cont;
  30   DOMAIN OF VALUE ARE  recu_secr,
  31                        classe_cont,
  32                        classe_etd_t;
  33 DEFINE MESSAGE  lettre_cont_recu_secr_MESS;
  34   KEYWORD ARE  "DOCUMENT";
  35   ATTRIBUTE IS  associe_a   "lettre_cont_ENT";
  36   CONSISTS OF  type_lettre_cont,
  37                explication_lettre_cont,
  38                contenu,
  39                echeance,
  40                statut,
  41                support;
```

Input Processor Source Listing

```
42 DEFINE MESSAGE  lettre_cont_classe_cont_MESS;
43   KEYWORD ARE  "DOCUMENT";
44   ATTRIBUTE IS  associe_a   "lettre_cont_ENT";
45   CONSISTS OF  type_lettre_cont,
46                explication_lettre_cont,
47                contenu,
48                echeance,
49                statut,
50                support;
51 DEFINE MESSAGE  lettre_cont_classe_etd_t_MESS;
52   KEYWORD ARE  "DOCUMENT";
53   ATTRIBUTE IS  associe_a   "lettre_cont_ENT";
54   CONSISTS OF  type_lettre_cont,
55                explication_lettre_cont,
56                contenu,
57                echeance,
58                statut,
59                support;
60 DEFINE RELATION  lettre_cont_REP_lettre_accep;
61   RELATES  lettre_cont_ENT
62     AS  a_pour_reponse
63     WITH CONNECTIVITY  "0_N";
64   RELATES  lettre_accep_ENT
65     AS  repond_a
66     WITH CONNECTIVITY  "0_N";
67 DEFINE RELATION  lettre_cont_REP_dde_inscr;
68   RELATES  lettre_cont_ENT
69     AS  a_pour_reponse
70     WITH CONNECTIVITY  "0_N";
71   RELATES  dde_inscr_ENT
72     AS  repond_a
73     WITH CONNECTIVITY  "0_N";
74 DEFINE RELATION  lettre_cont_EXP_etudiant;
75   CONSISTS OF  date_exp_lettre_cont;
76   RELATES  lettre_cont_ENT
77     AS  expedie_par
78     WITH CONNECTIVITY  "0_N";
79   RELATES  etudiant_ENT
80     AS  expedie
81     WITH CONNECTIVITY  "0_N";
82 DEFINE ELEMENT  date_exp_lettre_cont;
83   FORMAT IS  "99/99/9999";
84 DEFINE RELATION  lettre_cont_DEST_secret_adm;
85   RELATES  lettre_cont_ENT
86     AS  destine_a
87     WITH CONNECTIVITY  "0_N";
88   RELATES  secret_adm_ENT
89     AS  recoit
90     WITH CONNECTIVITY  "0_N";
```

90 lines with 51 statements.

A.3. Rapport d'ajout d'un fomulaire.

```
Interactive Design Approach     IDA2.0R0                    Page 1
                                FNDP - Namur VAX/VMS

                        Input Processor Source Listing
  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=FORMULAIRE.DSL
    SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

    1 DEFINE  ELEMENT  contenu;
    2    FORMAT IS  "texte";
    3 DEFINE  ELEMENT  echeance;
    4    FORMAT IS  "99/99/99";
    5 DEFINE  ELEMENT  statut;
    6    DOMAIN OF VALUE ARE  original,copie;
    7 DEFINE  ELEMENT  support;
    8    FORMAT IS  "alphabetique";
    9 DEFINE  GROUP sujet;
   10    CONSISTS OF  5 mot_cle;
   11 DEFINE  ELEMENT  mot_cle;
   12    FORMAT IS  "alphabetique";
   13 DEFINE  ELEMENT  corps;
   14    FORMAT IS  "texte";
   15 DEFINE  ELEMENT  paragraphes_statiques;
   16    FORMAT IS  "booleen";
   17 DEFINE ENTITY  dde_inscr_ENT;
   18   KEYWORD ARE  "FORMULAIRE";
   19   CONSISTS OF  type_dde_inscr,
   20                identifiant_dde_inscr,
   21                etat_dde_inscr,
   22                echeance,
   23                statut,
   24                support;
   25 DEFINE ELEMENT  type_dde_inscr;
   26   FORMAT IS "alphabetique";
   27   DOMAIN OF VALUE ARE  formulaire;
   28 DEFINE ELEMENT  identifiant_dde_inscr;
   29   FORMAT IS "alphabetique";
   30 DEFINE ELEMENT  etat_dde_inscr;
   31   FORMAT IS "alphabetique";
   32   DOMAIN OF VALUE ARE  a_remplir,
   33                        rempli,
   34                        a_corriger,
   35                        a_completer,
   36                        complet,
   37                        classe_inscr,
   38                        classe_etd_t;
   39 DEFINE MESSAGE  dde_inscr_a_remplir_MESS;
   40   KEYWORD ARE  "FORMULAIRE";
   41   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
   42   CONSISTS OF  type_dde_inscr,
   43                identifiant_dde_inscr,
   44                echeance,
   45                statut,
   46                support;
```

Input Processor Source Listing

```
47 DEFINE MESSAGE  dde_inscr_rempli_MESS;
48   KEYWORD ARE  "FORMULAIRE";
49   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
50   CONSISTS OF  type_dde_inscr,
51                identifiant_dde_inscr,
52                echeance,
53                statut,
54                support;
55 DEFINE MESSAGE  dde_inscr_a_corriger_MESS;
56   KEYWORD ARE  "FORMULAIRE";
57   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
58   CONSISTS OF  type_dde_inscr,
59                identifiant_dde_inscr,
60                echeance,
61                statut,
62                support;
63 DEFINE MESSAGE  dde_inscr_a_completer_MESS;
64   KEYWORD ARE  "FORMULAIRE";
65   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
66   CONSISTS OF  type_dde_inscr,
67                identifiant_dde_inscr,
68                echeance,
69                statut,
70                support;
71 DEFINE MESSAGE  dde_inscr_complet_MESS;
72   KEYWORD ARE  "FORMULAIRE";
73   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
74   CONSISTS OF  type_dde_inscr,
75                identifiant_dde_inscr,
76                echeance,
77                statut,
78                support;
79 DEFINE MESSAGE  dde_inscr_classe_inscr_MESS;
80   KEYWORD ARE  "FORMULAIRE";
81   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
82   CONSISTS OF  type_dde_inscr,
83                identifiant_dde_inscr,
84                echeance,
85                statut,
86                support;
87 DEFINE MESSAGE  dde_inscr_classe_etd_t_MESS;
88   KEYWORD ARE  "FORMULAIRE";
89   ATTRIBUTE IS  associe_a   "dde_inscr_ENT";
90   CONSISTS OF  type_dde_inscr,
91                identifiant_dde_inscr,
92                echeance,
93                statut,
94                support;
```

Input Processor Source Listing

```
 95 DEFINE RELATION  dde_inscr_EXP_secret_adm;
 96   CONSISTS OF  date_exp_dde_inscr;
 97   RELATES  dde_inscr_ENT
 98     AS  expedie_par
 99     WITH CONNECTIVITY  "0_N";
100   RELATES  secret_adm_ENT
101     AS  expedie
102     WITH CONNECTIVITY  "0_N";
103 DEFINE ELEMENT  date_exp_dde_inscr;
104   FORMAT IS  "99/99/9999";
105 DEFINE RELATION  dde_inscr_DEST_etudiant;
106   RELATES  dde_inscr_ENT
107     AS  destine_a
108     WITH CONNECTIVITY  "0_N";
109   RELATES  etudiant_ENT
110     AS  recoit
111     WITH CONNECTIVITY  "0_N";
112 DEFINE RELATION  dde_inscr_DEST_secret_centr;
113   RELATES  dde_inscr_ENT
114     AS  destine_a
115     WITH CONNECTIVITY  "0_N";
116   RELATES  secret_centr_ENT
117     AS  recoit
118     WITH CONNECTIVITY  "0_N";
119 DEFINE ENTITY  entete_ENT;
120   ATTRIBUTE IS  PARTIE_DE  "dde_inscr_ENT";
121   CONSISTS OF  etat_entete;
122 DEFINE ELEMENT  etat_entete;
123   FORMAT IS "alphabetique";
124   DOMAIN OF VALUE ARE  complet_t;
125 DEFINE RELATION  dde_inscr_COMP_entete;
126   CONSISTS OF  emplacement_entete;
127   RELATES  dde_inscr_ENT
128     AS  se_decompose_en
129     WITH CONNECTIVITY  "1_N";
130   RELATES  entete_ENT
131     AS  fait_partie_de
132     WITH CONNECTIVITY  "1_N";
133 DEFINE ELEMENT emplacement_entete;
134   DOMAIN OF VALUE ARE  "1_1";
135 DEFINE ENTITY  entete_txt_ENT;
136   ATTRIBUTE IS  ELT_TXT_DE  "entete_ENT";
137   CONSISTS OF  format_entete_txt,
138                contenu_entete_txt;
139 DEFINE ELEMENT  format_entete_txt;
140   DOMAIN OF VALUE ARE  "alphabetique";
141 DEFINE ELEMENT  contenu_entete_txt;
142   DOMAIN OF VALUE ARE  "Demande d'inscription";
```

Input Processor Source Listing

```
143 DEFINE RELATION  entete_COMT_entete_txt;
144   CONSISTS OF  emplacement_entete_txt;
145   RELATES  entete_ENT
146     AS  se_decompose_en
147     WITH CONNECTIVITY  "0_N";
148   RELATES  entete_txt_ENT
149     AS  fait_partie_de
150     WITH CONNECTIVITY  "1_N";
151 DEFINE ELEMENT  emplacement_entete_txt;
152   DOMAIN OF VALUE ARE  "1_1";
153 DEFINE ENTITY  identite_etd_ENT;
154   ATTRIBUTE IS  PARTIE_DE  "dde_inscr_ENT";
155   CONSISTS OF  etat_identite_etd;
156 DEFINE ELEMENT  etat_identite_etd;
157   FORMAT IS "alphabetique";
158   DOMAIN OF VALUE ARE  a_remplir,
159                        rempli,
160                        a_completer,
161                        a_corriger,
162                        complet_t;
163 DEFINE RELATION  dde_inscr_COMP_identite_etd;
164   CONSISTS OF  emplacement_identite_etd;
165   RELATES  dde_inscr_ENT
166     AS  se_decompose_en
167     WITH CONNECTIVITY  "1_N";
168   RELATES  identite_etd_ENT
169     AS  fait_partie_de
170     WITH CONNECTIVITY  "1_N";
171 DEFINE ELEMENT emplacement_identite_etd;
172   DOMAIN OF VALUE ARE  "2_1";
173 DEFINE ENTITY  remplisseur_ENT;
174   ATTRIBUTE IS  ELT_TXT_DE  "identite_etd_ENT";
175   CONSISTS OF  format_remplisseur,
176                contenu_remplisseur;
177 DEFINE ELEMENT  format_remplisseur;
178   DOMAIN OF VALUE ARE  "alphabetique";
179 DEFINE ELEMENT  contenu_remplisseur;
180   DOMAIN OF VALUE ARE  "a remplir par l'etudiant:";
181 DEFINE RELATION  identite_etd_COMT_remplisseur;
182   CONSISTS OF  emplacement_remplisseur;
183   RELATES  identite_etd_ENT
184     AS  se_decompose_en
185     WITH CONNECTIVITY  "0_N";
186   RELATES  remplisseur_ENT
187     AS  fait_partie_de
188     WITH CONNECTIVITY  "1_N";
189 DEFINE ELEMENT  emplacement_remplisseur;
190   DOMAIN OF VALUE ARE  "1_1";
```

Input Processor Source Listing

```
191 DEFINE ENTITY  identite_txt_ENT;
192   ATTRIBUTE IS  ELT_TXT_DE  "identite_etd_ENT";
193   CONSISTS OF  format_identite_txt,
194                contenu_identite_txt;
195 DEFINE ELEMENT  format_identite_txt;
196   DOMAIN OF VALUE ARE  "alphabetique";
197 DEFINE ELEMENT  contenu_identite_txt;
198   DOMAIN OF VALUE ARE  "identite:";
199 DEFINE RELATION  identite_etd_COMT_identite_txt;
200   CONSISTS OF  emplacement_identite_txt;
201   RELATES  identite_etd_ENT
202     AS  se_decompose_en
203     WITH CONNECTIVITY  "0_N";
204   RELATES  identite_txt_ENT
205     AS  fait_partie_de
206     WITH CONNECTIVITY  "1_N";
207 DEFINE ELEMENT  emplacement_identite_txt;
208   DOMAIN OF VALUE ARE  "2_1";
209 DEFINE ENTITY  nom_txt_ENT;
210   ATTRIBUTE IS  ELT_TXT_DE  "identite_etd_ENT";
211   CONSISTS OF  format_nom_txt,
212                contenu_nom_txt;
213 DEFINE ELEMENT  format_nom_txt;
214   DOMAIN OF VALUE ARE  "alphabetique";
215 DEFINE ELEMENT  contenu_nom_txt;
216   DOMAIN OF VALUE ARE  "nom:";
217 DEFINE RELATION  identite_etd_COMT_nom_txt;
218   CONSISTS OF  emplacement_nom_txt;
219   RELATES  identite_etd_ENT
220     AS  se_decompose_en
221     WITH CONNECTIVITY  "0_N";
222   RELATES  nom_txt_ENT
223     AS  fait_partie_de
224     WITH CONNECTIVITY  "1_N";
225 DEFINE ELEMENT  emplacement_nom_txt;
226   DOMAIN OF VALUE ARE  "3_1";
227 DEFINE ENTITY  nom_var_ENT;
228   ATTRIBUTE IS  ELT_VAR_DE  "identite_etd_ENT";
229   CONSISTS OF  format_nom_var,
230                val_def_nom_var;
231 DEFINE ELEMENT  format_nom_var;
232   DOMAIN OF VALUE ARE  "alphabetique";
233 DEFINE ELEMENT  val_def_nom_var;
234   DOMAIN OF VALUE ARE  "_____";
235 DEFINE RELATION nom_var_CORR_nom_txt;
236   RELATES  nom_var_ENT
237     AS  correspond_a
238     WITH CONNECTIVITY  "0_N";
239   RELATES  nom_txt_ENT
240     AS  correspond_a
241     WITH CONNECTIVITY  "0_1";
```

Input Processor Source Listing

```
242 DEFINE RELATION  identite_etd_COMV_nom_var;
243   CONSISTS OF  emplacement_nom_var;
244   RELATES  identite_etd_ENT
245     AS  se_decompose_en
246     WITH CONNECTIVITY  "0_N";
247   RELATES  nom_var_ENT
248     AS  fait_partie_de
249     WITH CONNECTIVITY  "1_N";
250 DEFINE ELEMENT emplacement_nom_var;
251   DOMAIN OF VALUE ARE  "3_5";
252 DEFINE ENTITY  prenom_txt_ENT;
253   ATTRIBUTE IS  ELT_TXT_DE  "identite_etd_ENT";
254   CONSISTS OF  format_prenom_txt,
255                contenu_prenom_txt;
256 DEFINE ELEMENT  format_prenom_txt;
257   DOMAIN OF VALUE ARE  "alphabetique";
258 DEFINE ELEMENT  contenu_prenom_txt;
259   DOMAIN OF VALUE ARE  "prenom:";
260 DEFINE RELATION  identite_etd_COMT_prenom_txt;
261   CONSISTS OF  emplacement_prenom_txt;
262   RELATES  identite_etd_ENT
263     AS  se_decompose_en
264     WITH CONNECTIVITY  "0_N";
265   RELATES  prenom_txt_ENT
266     AS  fait_partie_de
267     WITH CONNECTIVITY  "1_N";
268 DEFINE ELEMENT  emplacement_prenom_txt;
269   DOMAIN OF VALUE ARE  "4_1";
270 DEFINE ENTITY  prenom_var_ENT;
271   ATTRIBUTE IS  ELT_VAR_DE  "identite_etd_ENT";
272   CONSISTS OF  format_prenom_var,
273                val_def_prenom_var;
274 DEFINE ELEMENT  format_prenom_var;
275   DOMAIN OF VALUE ARE  "alphabetique";
276 DEFINE ELEMENT  val_def_prenom_var;
277   DOMAIN OF VALUE ARE  "_____";
278 DEFINE RELATION prenom_var_CORR_prenom_txt;
279   RELATES  prenom_var_ENT
280     AS  correspond_a
281     WITH CONNECTIVITY  "0_N";
282   RELATES  prenom_txt_ENT
283     AS  correspond_a
284     WITH CONNECTIVITY  "0_1";
285 DEFINE RELATION  identite_etd_COMV_prenom_var;
286   CONSISTS OF  emplacement_prenom_var;
287   RELATES  identite_etd_ENT
288     AS  se_decompose_en
289     WITH CONNECTIVITY  "0_N";
290   RELATES  prenom_var_ENT
291     AS  fait_partie_de
292     WITH CONNECTIVITY  "1_N";
```

Input Processor Source Listing

```
293 DEFINE ELEMENT emplacement_prenom_var;
294   DOMAIN OF VALUE ARE  "4_8";
295 DEFINE ENTITY  etudes_sup_ENT;
296   ATTRIBUTE IS  PARTIE_DE  "dde_inscr_ENT";
297   CONSISTS OF  etat_etudes_sup;
298 DEFINE ELEMENT  etat_etudes_sup;
299   FORMAT IS "alphabetique";
300   DOMAIN OF VALUE ARE  a_remplir,
301                        rempli,
302                        a_corriger,
303                        a_completer,
304                        complet_t;
305 DEFINE RELATION  dde_inscr_COMP_etudes_sup;
306   CONSISTS OF  emplacement_etudes_sup;
307   RELATES  dde_inscr_ENT
308     AS  se_decompose_en
309     WITH CONNECTIVITY  "1_N";
310   RELATES  etudes_sup_ENT
311     AS  fait_partie_de
312     WITH CONNECTIVITY  "1_N";
313 DEFINE ELEMENT emplacement_etudes_sup;
314   DOMAIN OF VALUE ARE  "7_1";
315 DEFINE ENTITY  etd_sup_txt_ENT;
316   ATTRIBUTE IS  ELT_TXT_DE  "etudes_sup_ENT";
317   CONSISTS OF  format_etd_sup_txt,
318                contenu_etd_sup_txt;
319 DEFINE ELEMENT  format_etd_sup_txt;
320   DOMAIN OF VALUE ARE  "alphabetique";
321 DEFINE ELEMENT  contenu_etd_sup_txt;
322   DOMAIN OF VALUE ARE  "etudes superieures:";
323 DEFINE RELATION  etudes_sup_COMT_etd_sup_txt;
324   CONSISTS OF  emplacement_etd_sup_txt;
325   RELATES  etudes_sup_ENT
326     AS  se_decompose_en
327     WITH CONNECTIVITY  "0_N";
328   RELATES  etd_sup_txt_ENT
329     AS  fait_partie_de
330     WITH CONNECTIVITY  "1_N";
331 DEFINE ELEMENT  emplacement_etd_sup_txt;
332   DOMAIN OF VALUE ARE  "1_1";
333 DEFINE ENTITY  annee_txt_ENT;
334   ATTRIBUTE IS  ELT_TXT_DE  "etudes_sup_ENT";
335   CONSISTS OF  format_annee_txt,
336                contenu_annee_txt;
337 DEFINE ELEMENT  format_annee_txt;
338   DOMAIN OF VALUE ARE  "alphabetique";
339 DEFINE ELEMENT  contenu_annee_txt;
340   DOMAIN OF VALUE ARE  "annee.";
```

Input Processor Source Listing

```
341 DEFINE RELATION  etudes_sup_COMT_annee_txt;
342   CONSISTS OF  emplacement_annee_txt;
343   RELATES  etudes_sup_ENT
344     AS  se_decompose_en
345     WITH CONNECTIVITY  "0_N";
346   RELATES  annee_txt_ENT
347     AS  fait_partie_de
348     WITH CONNECTIVITY  "1_N";
349 DEFINE ELEMENT  emplacement_annee_txt;
350   DOMAIN OF VALUE ARE  "2_1";
351 DEFINE ENTITY  annee_var_ENT;
352   ATTRIBUTE IS  ELT_VAR_DE  "etudes_sup_ENT";
353   CONSISTS OF  format_annee_var,
354                val_def_annee_var;
355 DEFINE ELEMENT  format_annee_var;
356   DOMAIN OF VALUE ARE  "aaaa/aaaa";
357 DEFINE ELEMENT  val_def_annee_var;
358   DOMAIN OF VALUE ARE  "____/____";
359 DEFINE RELATION annee_var_CORR_annee_txt;
360   RELATES  annee_var_ENT
361     AS  correspond_a
362     WITH CONNECTIVITY  "0_N";
363   RELATES  annee_txt_ENT
364     AS  correspond_a
365     WITH CONNECTIVITY  "0_1";
366 DEFINE RELATION  etudes_sup_COMV_annee_var;
367   CONSISTS OF  emplacement_annee_var;
368   RELATES  etudes_sup_ENT
369     AS  se_decompose_en
370     WITH CONNECTIVITY  "0_N";
371   RELATES  annee_var_ENT
372     AS  fait_partie_de
373     WITH CONNECTIVITY  "1_N";
374 DEFINE ELEMENT emplacement_annee_var;
375   DOMAIN OF VALUE ARE  "3_1",
376                        "4_1",
377                        "5_1";
378 DEFINE ENTITY  etudes_txt_ENT;
379   ATTRIBUTE IS  ELT_TXT_DE  "etudes_sup_ENT";
380   CONSISTS OF  format_etudes_txt,
381                contenu_etudes_txt;
382 DEFINE ELEMENT  format_etudes_txt;
383   DOMAIN OF VALUE ARE  "alphabetique";
384 DEFINE ELEMENT  contenu_etudes_txt;
385   DOMAIN OF VALUE ARE  "etudes.";
```

Input Processor Source Listing

```
386 DEFINE RELATION  etudes_sup_COMT_etudes_txt;
387   CONSISTS OF  emplacement_etudes_txt;
388   RELATES  etudes_sup_ENT
389     AS  se_decompose_en
390     WITH CONNECTIVITY  "0_N";
391   RELATES  etudes_txt_ENT
392     AS  fait_partie_de
393     WITH CONNECTIVITY  "1_N";
394 DEFINE ELEMENT  emplacement_etudes_txt;
395   DOMAIN OF VALUE ARE  "2_20";
396 DEFINE ENTITY  etudes_var_ENT;
397   ATTRIBUTE IS  ELT_VAR_DE  "etudes_sup_ENT";
398   CONSISTS OF  format_etudes_var,
399                val_def_etudes_var;
400 DEFINE ELEMENT  format_etudes_var;
401   DOMAIN OF VALUE ARE  "alphabetique";
402 DEFINE ELEMENT  val_def_etudes_var;
403   DOMAIN OF VALUE ARE  "_____";
404 DEFINE RELATION etudes_var_CORR_etudes_txt;
405   RELATES  etudes_var_ENT
406     AS  correspond_a
407     WITH CONNECTIVITY  "0_N";
408   RELATES  etudes_txt_ENT
409     AS  correspond_a
410     WITH CONNECTIVITY  "0_1";
411 DEFINE RELATION  etudes_sup_COMV_etudes_var;
412   CONSISTS OF  emplacement_etudes_var;
413   RELATES  etudes_sup_ENT
414     AS  se_decompose_en
415     WITH CONNECTIVITY  "0_N";
416   RELATES  etudes_var_ENT
417     AS  fait_partie_de
418     WITH CONNECTIVITY  "1_N";
419 DEFINE ELEMENT emplacement_etudes_var;
420   DOMAIN OF VALUE ARE  "3_20",
421                        "4_20",
422                        "5_20";
```

422 lines with 271 statements.
No diagnostics.

A.4. Rapport d'ajout d'un dossier.

```
Interactive Design Approach    IDA2.0R0                    Page 1
                               FNDP - Namur VAX/VMS

                          Input Processor Source Listing

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=DOSSIER.DSL
    SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

    1 DEFINE  ELEMENT  contenu;
    2    FORMAT IS  "texte";
    3 DEFINE  ELEMENT  echeance;
    4    FORMAT IS  "99/99/99";
    5 DEFINE  ELEMENT  statut;
    6    DOMAIN OF VALUE ARE  original,copie;
    7 DEFINE  ELEMENT  support;
    8    FORMAT IS  "alphabetique";
    9 DEFINE  GROUP sujet;
   10    CONSISTS OF  5 mot_cle;
   11 DEFINE  ELEMENT  mot_cle;
   12    FORMAT IS  "alphabetique";
   13 DEFINE  ELEMENT  corps;
   14    FORMAT IS  "texte";
   15 DEFINE  ELEMENT  paragraphes_statiques;
   16    FORMAT IS  "booleen";
   17 DEFINE ENTITY  reponse_ENT;
   18   KEYWORD ARE  "DOSSIER";
   19   CONSISTS OF  identifiant_reponse,
   20                etat_reponse,
   21                echeance,
   22                support;
   23 DEFINE ELEMENT  identifiant_reponse;
   24   FORMAT IS  "alphabetique";
   25 DEFINE ELEMENT  etat_reponse;
   26   DOMAIN OF VALUE ARE  envoye_etd_t;
   27 DEFINE MESSAGE  reponse_envoye_etd_t_MESS;
   28   KEYWORD ARE  "DOSSIER";
   29   ATTRIBUTE IS  associe_a   "reponse_ENT";
   30   CONSISTS OF  identifiant_reponse,
   31                echeance,
   32                support;
   33 DEFINE RELATION  reponse_EXP_secret_adm;
   34   CONSISTS OF  date_exp_reponse;
   35   RELATES  reponse_ENT
   36     AS  expedie_par
   37     WITH CONNECTIVITY  "0_N";
   38   RELATES  secret_adm_ENT
   39     AS  expedie
   40     WITH CONNECTIVITY  "0_N";
   41 DEFINE ELEMENT  date_exp_reponse;
   42   FORMAT IS  "99/99/9999";
```

Input Processor Source Listing

```
43 DEFINE RELATION  reponse_DEST_etudiant;
44   RELATES  reponse_ENT
45     AS  destine_a
46     WITH CONNECTIVITY  "0_N";
47   RELATES  etudiant_ENT
48     AS  recoit
49     WITH CONNECTIVITY  "0_N";
50 DEFINE RELATION  reponse_CONS_lettre_accep;
51   RELATES  reponse_ENT
52     AS  contient
53     WITH CONNECTIVITY  "0_N";
54   RELATES  lettre_accep_ENT
55     AS  est_contenu_dans
56     WITH CONNECTIVITY  "0_1";
57 DEFINE RELATION  reponse_CONS_dde_inscr;
58   RELATES  reponse_ENT
59     AS  contient
60     WITH CONNECTIVITY  "0_N";
61   RELATES  dde_inscr_ENT
62     AS  est_contenu_dans
63     WITH CONNECTIVITY  "0_1";

63 lines with 42 statements.
No diagnostics.
```

A.5. Rapport d'ajout d'un fichier.

```
Interactive Design Approach     IDA2.0R0                    Page 1
                                FNDP - Namur VAX/VMS

                          Input Processor Source Listing

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=FICHIER.DSL
    SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

    1 DEFINE  ELEMENT  contenu;
    2    FORMAT IS  "texte";
    3 DEFINE  ELEMENT  echeance;
    4    FORMAT IS  "99/99/99";
    5 DEFINE  ELEMENT  statut;
    6    DOMAIN OF VALUE ARE  original,copie;
    7 DEFINE  ELEMENT  support;
    8    FORMAT IS  "alphabetique";
    9 DEFINE  GROUP sujet;
   10    CONSISTS OF  5 mot_cle;
   11 DEFINE  ELEMENT  mot_cle;
   12    FORMAT IS  "alphabetique";
   13 DEFINE  ELEMENT  corps;
   14    FORMAT IS  "texte";
   15 DEFINE  ELEMENT  paragraphes_statiques;
   16    FORMAT IS  "booleen";
   17 DEFINE ENTITY  fich_contact_ENT;
   18   KEYWORD ARE  "FICHIER";
   19   CONSISTS OF  etat_fich_contact,
   20                support;
   21 DEFINE ELEMENT  etat_fich_contact;
   22   DOMAIN OF VALUE ARE  ouvert,
   23                        vide_t;
   24 DEFINE RELATION  fich_contact_CONS_lettre_cont;
   25   RELATES  fich_contact_ENT
   26     AS  contient
   27     WITH CONNECTIVITY  "0_N";
   28   RELATES  lettre_cont_ENT
   29     AS  est_contenu_dans
   30     WITH CONNECTIVITY  "0_1";

30 lines with 24 statements.
No diagnostics.
```

A.5. Rapport d'ajout d'une pile.

```
Interactive Design Approach    IDA2.0RO                    Page 1
                               FNDP - Namur VAX/VMS

                          Input Processor Source Listing

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  INPUT=PILE.DSL
  SOURCE-LISTING  NOCROSS-REFERENCE  UPDATE  DATA-BASE-REFERENCE

    1 DEFINE  ELEMENT  contenu;
    2    FORMAT IS  "texte";
    3 DEFINE  ELEMENT  echeance;
    4    FORMAT IS  "99/99/99";
    5 DEFINE  ELEMENT  statut;
    6    DOMAIN OF VALUE ARE  original,copie;
    7 DEFINE  ELEMENT  support;
    8    FORMAT IS  "alphabetique";
    9 DEFINE  GROUP sujet;
   10    CONSISTS OF  5 mot_cle;
   11 DEFINE  ELEMENT  mot_cle;
   12    FORMAT IS  "alphabetique";
   13 DEFINE  ELEMENT  corps;
   14    FORMAT IS  "texte";
   15 DEFINE  ELEMENT  paragraphes_statiques;
   16    FORMAT IS  "booleen";
   17 DEFINE ENTITY  papiers_dus_ENT;
   18   KEYWORD ARE  "PILE";
   19   CONSISTS OF  etat_papiers_dus,
   20                support;
   21 DEFINE ELEMENT  etat_papiers_dus;
   22   DOMAIN OF VALUE ARE  non_vide,
   23                        vide_t;

23 lines with 21 statements.
No diagnostics.
```

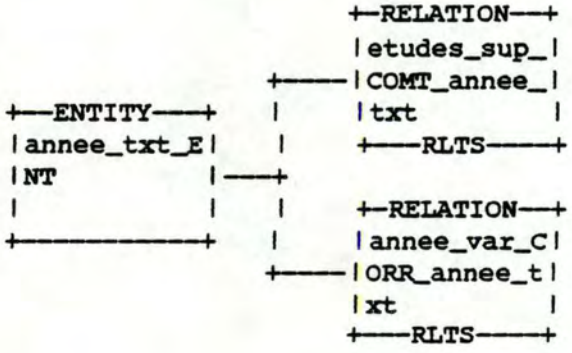
B. Les rapports documentaires de DSL-SPEC.

B.1. Le rapport EXTENDED-PICTURE

Il présente, sous forme graphique, pour une liste de noms d'objets donnés en entrée, un réseau de noms reliés aux précédents par une relation.

Commande: "EP ALL FILE=OBJETS.DTA P ARW BOX RELATION=RELATED-BY"

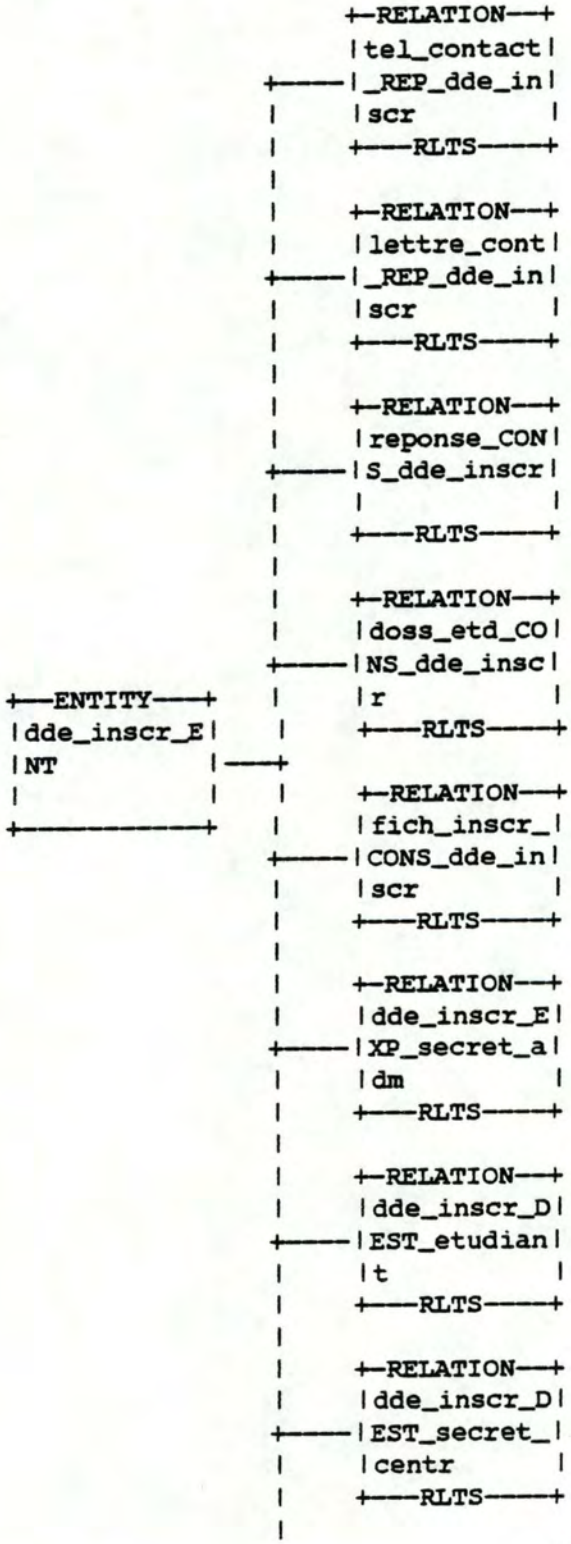
```
Interactive Design Approach    IDA2.0R0                    Page 1
                              FNDP - Namur VAX/VMS

                                Extended Picture

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBP  FILE=OBJETS.DTA
   OBJECTS=ALL  RELATIONS=RELTD  LINKS=50  PRINT  BOXES  ALL
   HORIZONTAL-BOXES=4  VERTICAL-BOXES=8  NOARROWS  NOPLOT

NAME=annee_txt_ENT                                  PAGE 1 OF 1
```



Interactive Design Approach IDA2.0RO Page 2
FNDP - Namur VAX/VMS

Extended Picture

NAME=dde_inscr_ENT PAGE 1 OF 2

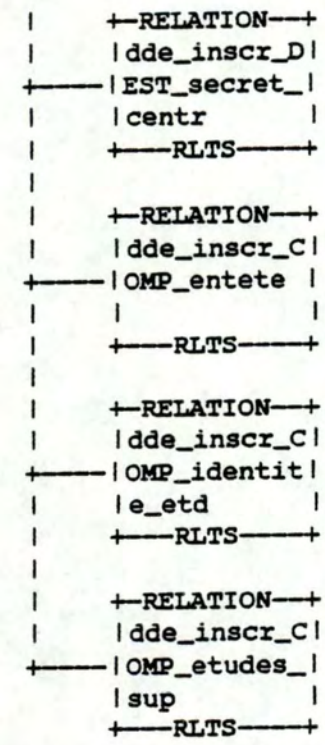


TO PAGE 2

Interactive Design Approach IDA2.0R0 Page 3
FNDP - Namur VAX/VMS

Extended Picture

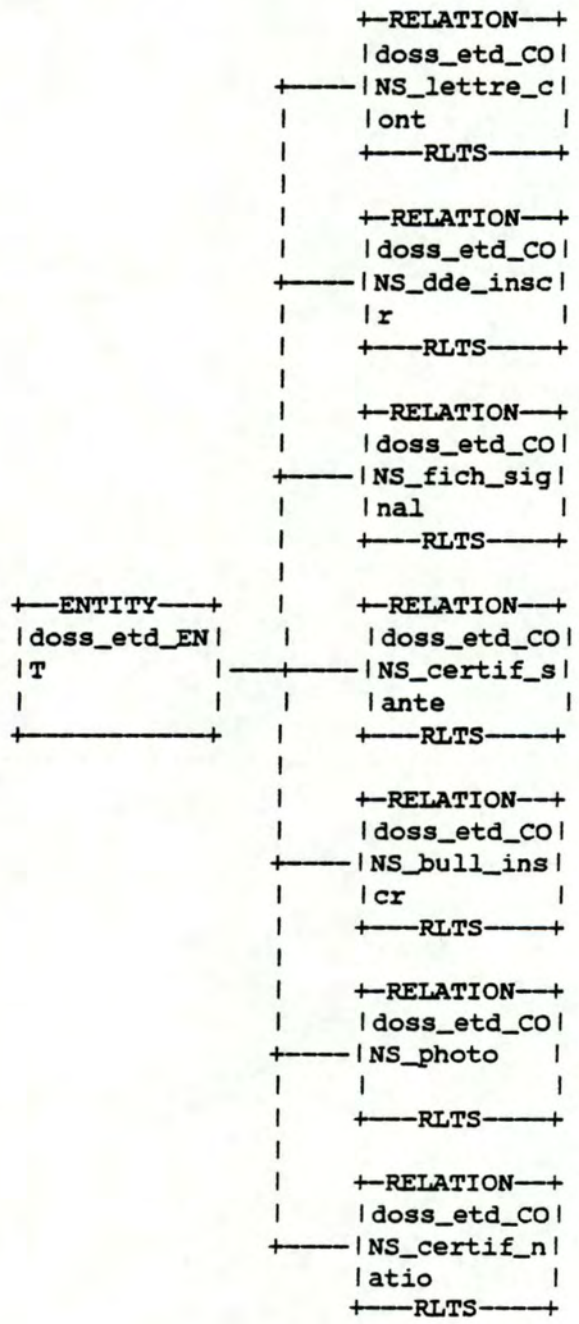
NAME=dde_inscr_ENT FROM PAGE 1 PAGE 2 OF 2



Interactive Design Approach IDA2.0RO Page 4
FNDP - Namur VAX/VMS

Extended Picture

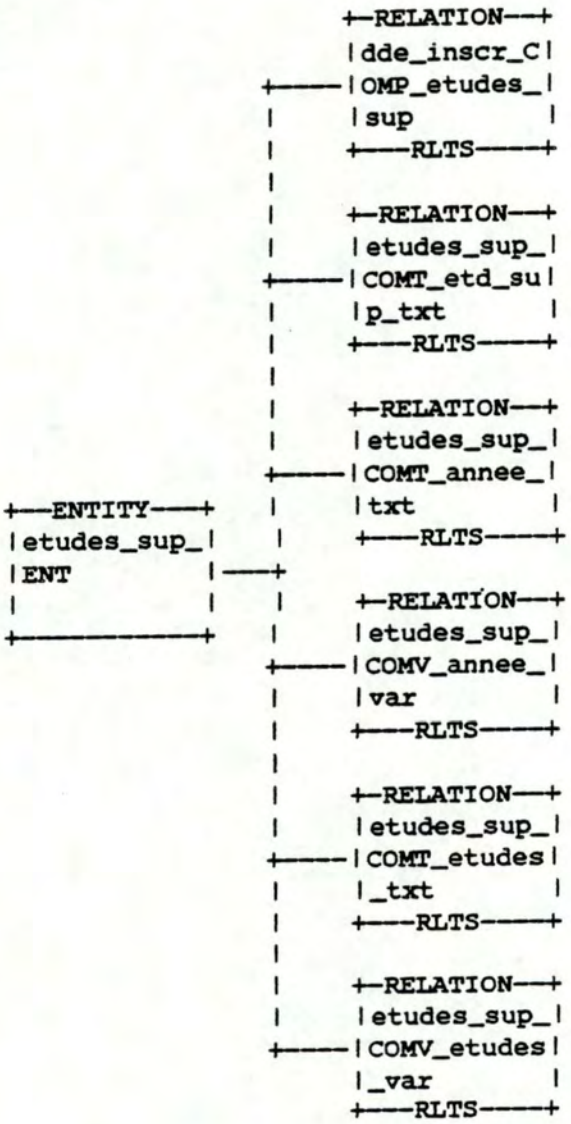
NAME=doss_etd_ENT PAGE 1 OF 1



Interactive Design Approach IDA2.0R0 Page 5
FNDP - Namur VAX/VMS

Extended Picture

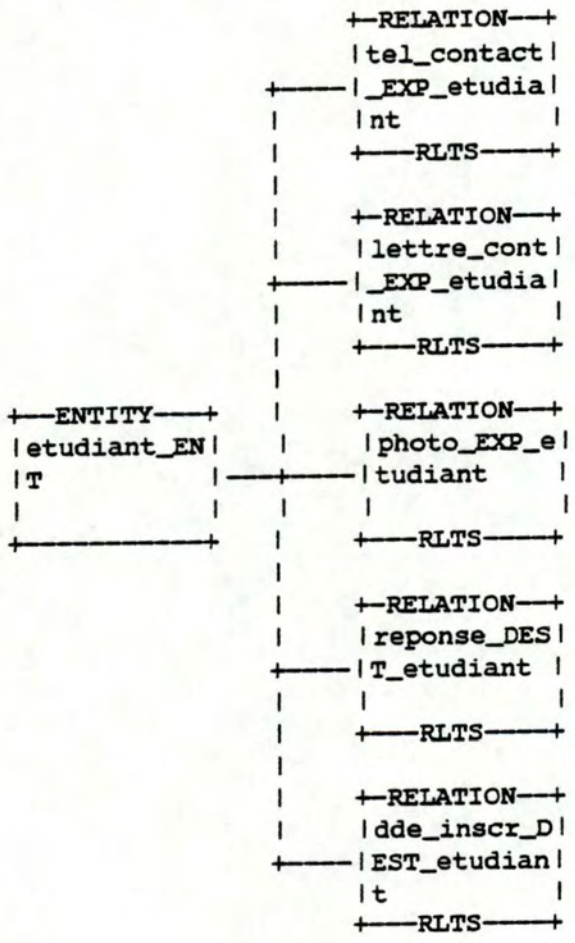
NAME=etudes_sup_ENT PAGE 1 OF 1



Interactive Design Approach IDA2.0RO Page 6
FNDP - Namur VAX/VMS

Extended Picture

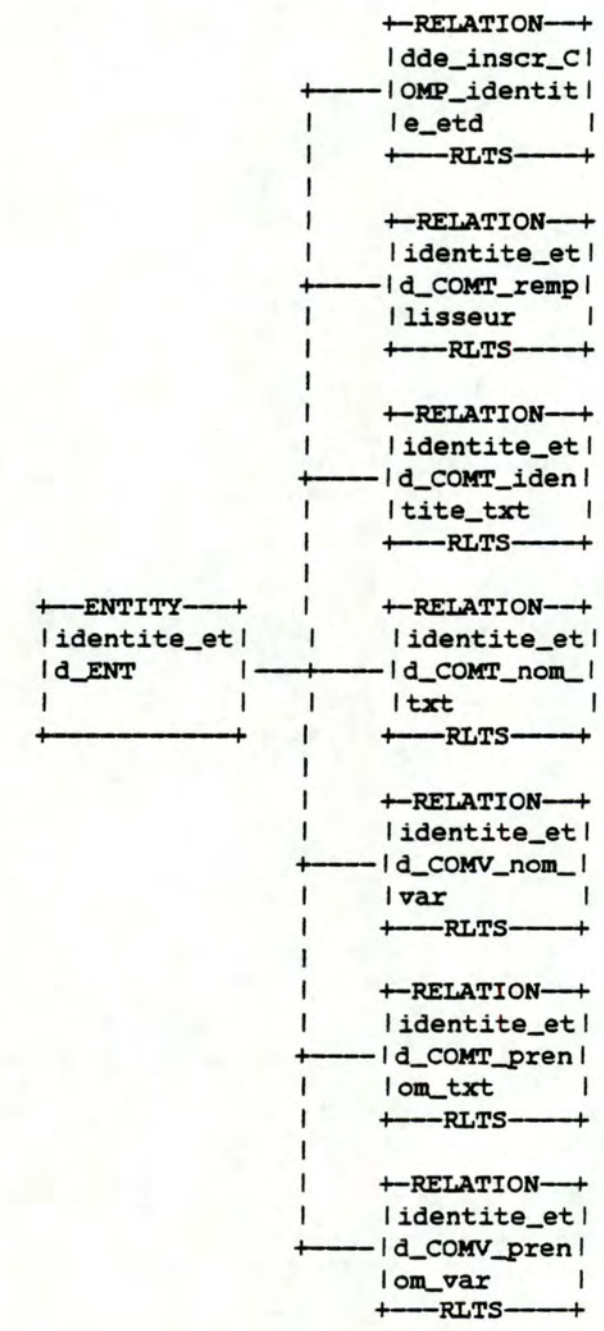
NAME=etudiant_ENT PAGE 1 OF 1



Interactive Design Approach IDA2.0R0 Page 7
FNDP - Namur VAX/VMS

Extended Picture

NAME=identite_etd_ENT PAGE 1 OF 1



B.2. Le rapport FORMATTED-STATEMENT.

Il présente, à l'aide de phrases DSL, et pour une liste d'objets donnée par l'utilisateur, toutes les informations stockées dans la base de données.

Exemple: information sur l'objet de bureau "tel_contact"

Commande: "FS P NAME=tel_contact"

```
Interactive Design Approach     IDA2.0R0                        Page 1
                               FNDP - Namur VAX/VMS

                               Formatted Statements

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  NAME=tel_contact_ENT
   PRINT  NOPUNCH  HSMARG=0  HNMARG=35  SMARG=5  NMARG=25  TMARG=0
   SRMARG=80 ONE-PER-LINE  DLC-COMMENT  NONEW-PAGE

    1
    2 DEFINE ENTITY                          tel_contact_ENT;
    3 # Last changed - Aug 06, 1986  11:04:55
    4       CONSISTS OF          type_tel_contact;
    5       CONSISTS OF          explication_tel_contact;
    6       CONSISTS OF          contenu;
    7       CONSISTS OF          etat_tel_contact;
    8       CONSISTS OF          echeance;
    9       CONSISTS OF          statut;
   10       CONSISTS OF          support;
   11       KEYWORD              'MSGE';
   12       RELATED BY           tel_contact_REP_lettre_accep AS
   13                            a_pour_reponse WITH CONNECTIVITY
   14                            '0_N';
   15       RELATED BY           tel_contact_REP_dde_inscr AS
   16                            a_pour_reponse WITH CONNECTIVITY
   17                            '0_N';
   18       RELATED BY           tel_contact_EXP_etudiant AS
   19                            expedie_par WITH CONNECTIVITY
   20                            '0_N';
   21       RELATED BY           tel_contact_DEST_secret_adm AS
   22                            destine_a WITH CONNECTIVITY
   23                            '0_N';
   24
```

B.3. Le rapport NAME-LIST

Il présente une liste des objets de la base de données en associant, pour chacun d'eux, son type.

Commande: "NL NSYN"

Interactive Design Approach IDA2.0RO Page 1
FNDP - Namur VAX/VMS

Name List

Parameters: LANGUAGE=DSL DB=DONNEES.DBF TYPE NOSYNONYM NODATE-LAST-CHANGED BYTYPE

	Name	Type
1	statut	***Undefined***
2	ELT_TXT_DE	ATTRIBUTE
3	ELT_VAR_DE	ATTRIBUTE
4	PARTIE_DE	ATTRIBUTE
5	associe_a	ATTRIBUTE
6	contenu	ELEMENT
7	contenu_annee_txt	ELEMENT
8	contenu_entete_txt	ELEMENT
9	contenu_etd_sup_txt	ELEMENT
10	contenu_etudes_txt	ELEMENT
11	contenu_identite_txt	ELEMENT
12	contenu_nom_txt	ELEMENT
13	contenu_prenom_txt	ELEMENT
14	contenu_remplisseur	ELEMENT
15	corps	ELEMENT
16	date_exp_dde_inscr	ELEMENT
17	date_exp_lettre_cont	ELEMENT
18	date_exp_photo	ELEMENT
19	date_exp_rentree	ELEMENT
20	date_exp_reponse	ELEMENT
21	date_exp_tel_contact	ELEMENT
22	echeance	ELEMENT
23	emplacement_annee_txt	ELEMENT
24	emplacement_annee_var	ELEMENT
25	emplacement_entete	ELEMENT
26	emplacement_entete_txt	ELEMENT
27	emplacement_etd_sup_txt	ELEMENT
28	emplacement_etudes_sup	ELEMENT
29	emplacement_etudes_txt	ELEMENT
30	emplacement_etudes_var	ELEMENT
31	emplacement_identite_etd	ELEMENT
32	emplacement_identite_txt	ELEMENT
33	emplacement_nom_txt	ELEMENT
34	emplacement_nom_var	ELEMENT
35	emplacement_prenom_txt	ELEMENT
36	emplacement_prenom_var	ELEMENT

Name List

	Name	Type
37	emplacement_remplisseur	ELEMENT
38	etat_dde_inscr	ELEMENT
39	etat_doss_etd	ELEMENT
40	etat_entete	ELEMENT
41	etat_etudes_sup	ELEMENT
42	etat_fich_contact	ELEMENT
43	etat_fich_inscr	ELEMENT
44	etat_identite_etd	ELEMENT
45	etat_lettre_cont	ELEMENT
46	etat_papiers_dus	ELEMENT
47	etat_photo	ELEMENT
48	etat_rentree	ELEMENT
49	etat_reponse	ELEMENT
50	etat_tel_contact	ELEMENT
51	explication_lettre_cont	ELEMENT
52	explication_photo	ELEMENT
53	explication_rentree	ELEMENT
54	explication_tel_contact	ELEMENT
55	format_annee_txt	ELEMENT
56	format_annee_var	ELEMENT
57	format_entete_txt	ELEMENT
58	format_etd_sup_txt	ELEMENT
59	format_etudes_txt	ELEMENT
60	format_etudes_var	ELEMENT
61	format_identite_txt	ELEMENT
62	format_nom_txt	ELEMENT
63	format_nom_var	ELEMENT
64	format_prenom_txt	ELEMENT
65	format_prenom_var	ELEMENT
66	format_remplisseur	ELEMENT
67	identifiant_dde_inscr	ELEMENT
68	identifiant_doss_etd	ELEMENT
69	identifiant_reponse	ELEMENT
70	mot_cle	ELEMENT
71	paragraphes_statiques	ELEMENT
72	statut	ELEMENT
73	support	ELEMENT
74	type_dde_inscr	ELEMENT
75	type_lettre_cont	ELEMENT
76	type_photo	ELEMENT
77	type_rentree	ELEMENT
78	type_tel_contact	ELEMENT
79	val_def_annee_var	ELEMENT
80	val_def_etudes_var	ELEMENT
81	val_def_nom_var	ELEMENT
82	val_def_prenom_var	ELEMENT
83	annee_txt_ENT	ENTITY
84	annee_var_ENT	ENTITY
85	bull_inscr_ENT	ENTITY

Name List

	Name	Type
86	certif_natio_ENT	ENTITY
87	certif_sante_ENT	ENTITY
88	dde_inscr_ENT	ENTITY
89	doss_etd_ENT	ENTITY
90	entete_ENT	ENTITY
91	entete_txt_ENT	ENTITY
92	environ_ENT	ENTITY
93	etd_sup_txt_ENT	ENTITY
94	etudes_sup_ENT	ENTITY
95	etudes_txt_ENT	ENTITY
96	etudes_var_ENT	ENTITY
97	etudiant_ENT	ENTITY
98	fich_contact_ENT	ENTITY
99	fich_inscr_ENT	ENTITY
100	fich_signal_ENT	ENTITY
101	identite_etd_ENT	ENTITY
102	identite_txt_ENT	ENTITY
103	lettre_accep_ENT	ENTITY
104	lettre_cont_ENT	ENTITY
105	nom_txt_ENT	ENTITY
106	nom_var_ENT	ENTITY
107	papiers_dus_ENT	ENTITY
108	photo_ENT	ENTITY
109	prenom_txt_ENT	ENTITY
110	prenom_var_ENT	ENTITY
111	remplisseur_ENT	ENTITY
112	rentree_ENT	ENTITY
113	reponse_ENT	ENTITY
114	secret_adm_ENT	ENTITY
115	secret_centr_ENT	ENTITY
116	tel_contact_ENT	ENTITY
117	sujet	GROUP
118	dde_inscr_a_completer_MESS	MESSAGE
119	dde_inscr_a_corriger_MESS	MESSAGE
120	dde_inscr_a_remplir_MESS	MESSAGE
121	dde_inscr_classe_etd_t_MESS	MESSAGE
122	dde_inscr_classe_inscr_MESS	MESSAGE
123	dde_inscr_complet_MESS	MESSAGE
124	dde_inscr_rempli_MESS	MESSAGE
125	doss_etd_complet_t_MESS	MESSAGE
126	doss_etd_cree_MESS	MESSAGE
127	lettre_cont_classe_cont_MESS	MESSAGE
128	lettre_cont_classe_etd_t_MESS	MESSAGE
129	lettre_cont_recu_secr_MESS	MESSAGE
130	photo_classe_dus_MESS	MESSAGE
131	photo_classe_etd_t_MESS	MESSAGE
132	rentree_en_attente_MESS	MESSAGE
133	rentree_survenu_t_MESS	MESSAGE

Name List

	Name	Type
134	reponse_envoye_etd_t_MESS	MESSAGE
135	tel_contact_envoye_secr_MESS	MESSAGE
136	tel_contact_traite_t_MESS	MESSAGE
137	annee_var_CORR_annee_txt	RELATION
138	dde_inscr_COMP_entete	RELATION
139	dde_inscr_COMP_etudes_sup	RELATION
140	dde_inscr_COMP_identite_etd	RELATION
141	dde_inscr_DEST_etudiant	RELATION
142	dde_inscr_DEST_secret_centr	RELATION
143	dde_inscr_EXP_secret_adm	RELATION
144	doss_etd_CONS_bull_inscr	RELATION
145	doss_etd_CONS_certif_natio	RELATION
146	doss_etd_CONS_certif_sante	RELATION
147	doss_etd_CONS_dde_inscr	RELATION
148	doss_etd_CONS_fich_signal	RELATION
149	doss_etd_CONS_lettre_cont	RELATION
150	doss_etd_CONS_photo	RELATION
151	entete_COMT_entete_txt	RELATION
152	etudes_sup_COMT_annee_txt	RELATION
153	etudes_sup_COMT_etd_sup_txt	RELATION
154	etudes_sup_COMT_etudes_txt	RELATION
155	etudes_sup_COMV_annee_var	RELATION
156	etudes_sup_COMV_etudes_var	RELATION
157	etudes_var_CORR_etudes_txt	RELATION
158	fich_contact_CONS_lettre_cont	RELATION
159	fich_inscr_CONS_dde_inscr	RELATION
160	identite_etd_COMT_identite_txt	RELATION
161	identite_etd_COMT_nom_txt	RELATION
162	identite_etd_COMT_prenom_txt	RELATION
163	identite_etd_COMT_remplisseur	RELATION
164	identite_etd_COMV_nom_var	RELATION
165	identite_etd_COMV_prenom_var	RELATION
166	lettre_cont_DEST_secret_adm	RELATION
167	lettre_cont_EXP_etudiant	RELATION
168	lettre_cont_REP_dde_inscr	RELATION
169	lettre_cont_REP_lettre_accep	RELATION
170	nom_var_CORR_nom_txt	RELATION
171	photo_DEST_secret_adm	RELATION
172	photo_EXP_etudiant	RELATION
173	prenom_var_CORR_prenom_txt	RELATION
174	rentree_DEST_secret_adm	RELATION
175	rentree_EXP_environ	RELATION
176	reponse_CONS_dde_inscr	RELATION
177	reponse_CONS_lettre_accep	RELATION
178	reponse_DEST_etudiant	RELATION
179	reponse_EXP_secret_adm	RELATION
180	tel_contact_DEST_secret_adm	RELATION
181	tel_contact_EXP_etudiant	RELATION

Name List

	Name	Type
182	tel_contact_REP_dde_inscr	RELATION
183	tel_contact_REP_lettre_accep	RELATION
184	a_pour_reponse	ROLE
185	contient	ROLE
186	correspond_a	ROLE
187	destine_a	ROLE
188	est_contenu_dans	ROLE
189	expedie	ROLE
190	expedie_par	ROLE
191	fait_partie_de	ROLE
192	recoit	ROLE
193	repond_a	ROLE
194	se_decompose_en	ROLE
195	a_completer	SYSTEM-PARAMETER
196	a_corriger	SYSTEM-PARAMETER
197	a_remplir	SYSTEM-PARAMETER
198	autre	SYSTEM-PARAMETER
199	calendrier	SYSTEM-PARAMETER
200	classe_cont	SYSTEM-PARAMETER
201	classe_dus	SYSTEM-PARAMETER
202	classe_etd_t	SYSTEM-PARAMETER
203	classe_inscr	SYSTEM-PARAMETER
204	cloture	SYSTEM-PARAMETER
205	complet	SYSTEM-PARAMETER
206	complet_t	SYSTEM-PARAMETER
207	copie	SYSTEM-PARAMETER
208	cree	SYSTEM-PARAMETER
209	en_attente	SYSTEM-PARAMETER
210	envoye_etd_t	SYSTEM-PARAMETER
211	envoye_secr	SYSTEM-PARAMETER
212	formulaire	SYSTEM-PARAMETER
213	lettre	SYSTEM-PARAMETER
214	non_vide	SYSTEM-PARAMETER
215	original	SYSTEM-PARAMETER
216	ouvert	SYSTEM-PARAMETER
217	photographie	SYSTEM-PARAMETER

B.4. Le rapport STRUCTURED-FS

Il présente, sous forme de liste décalée, pour une liste de noms d'objets en entrée, une hiérarchie de noms reliés aux précédents par un nom de relation.

Exemple: fichier des objets reliés aux autres par la relation "CONSITS OF"

Commande: "STFS FILE=OBJETS.DTA REL=CSTS-OF"

```
Interactive Design Approach     IDA2.0R0                    Page 1
                                FNDP - Namur VAX/VMS

                        Structured Formated Statement Report

  Parameters:   LANGUAGE=DSL  DB=DONNEES.DBF  FILE=OBJETS.DTA
    NOPUNCH  OBJECTS=ALL  RELATIONS=CSTS-OF  PRINT  INDENT=2
    LEVELS=50  TYPES-MARGIN=36  RELATIONS-MARGIN=48  LINE-NUMBERS
    LEVEL-NUMBERS  STATISTICS  NONEW-PAGE  OBJECT-TYPES
    ROW-ORDER=STANDARD  COLUMN-ORDER=STANDARD  NOLOWEST-LEVEL-ONLY
    NORELATION-MATRIX   NOCOMPRESS-RELATION-MATRIX


  1    1 annee_txt_ENT              <ENTITY>
  2     2  format_annee_txt         <ELEMENT>
                                        CONTAINED IN annee_txt_ENT
  3     2  contenu_annee_txt        <ELEMENT>
                                        CONTAINED IN annee_txt_ENT



 LEVEL COUNT   LEVEL COUNT
     1     1       2     2


  1    1 annee_var_ENT              <ENTITY>
  2     2  format_annee_var         <ELEMENT>
                                        CONTAINED IN annee_var_ENT
  3     2  val_def_annee_var        <ELEMENT>
                                        CONTAINED IN annee_var_ENT
```

Structured Formated Statement Report

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1    1 bull_inscr_ENT              <ENTITY>




LEVEL COUNT
    1     1

  1    1 certif_natio_ENT            <ENTITY>



LEVEL COUNT
    1     1

  1    1 certif_sante_ENT            <ENTITY>



LEVEL COUNT
    1     1

  1    1 dde_inscr_ENT               <ENTITY>
  2     2  type_dde_inscr            <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
  3     2  identifiant_dde_inscr     <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
  4     2  etat_dde_inscr            <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
  5     2  echeance                  <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
  6     2  statut                    <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
  7     2  support                   <ELEMENT>
                                        CONTAINED IN dde_inscr_ENT
```

Structured Formated Statement Report

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     6


  1    1 doss_etd_ENT              <ENTITY>
  2     2  identifiant_doss_etd    <ELEMENT>
                                       CONTAINED IN doss_etd_ENT
  3     2  etat_doss_etd           <ELEMENT>
                                       CONTAINED IN doss_etd_ENT
  4     2  echeance                <ELEMENT>
                                       CONTAINED IN doss_etd_ENT
  5     2  support                 <ELEMENT>
                                       CONTAINED IN doss_etd_ENT


LEVEL COUNT   LEVEL COUNT
    1     1       2     4
  1    1 entete_ENT                <ENTITY>
  2     2  etat_entete             <ELEMENT>
                                         CONTAINED IN entete_ENT


LEVEL COUNT   LEVEL COUNT
    1     1       2     1


  1    1 entete_txt_ENT            <ENTITY>
  2     2  format_entete_txt       <ELEMENT>
                                     CONTAINED IN entete_txt_ENT
  3     2  contenu_entete_txt      <ELEMENT>
                                     CONTAINED IN entete_txt_ENT


LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1    1 environ_ENT               <ENTITY>
```

Structured Formated Statement Report

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     1


  1   1 etudes_txt_ENT            <ENTITY>
  2    2  format_etudes_txt       <ELEMENT>
                                    CONTAINED IN etudes_txt_ENT
  3    2  contenu_etudes_txt      <ELEMENT>
                                    CONTAINED IN etudes_txt_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1   1 etudes_var_ENT            <ENTITY>
  2    2  format_etudes_var       <ELEMENT>
                                    CONTAINED IN etudes_var_ENT
  3    2  val_def_etudes_var      <ELEMENT>
                                    CONTAINED IN etudes_var_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1   1 etudiant_ENT              <ENTITY>
```

```
LEVEL COUNT
    1     1


  1   1 fich_contact_ENT          <ENTITY>
  2    2  etat_fich_contact       <ELEMENT>
                                   CONTAINED IN fich_contact_ENT
  3    2  support                 <ELEMENT>
                                   CONTAINED IN fich_contact_ENT
```

Structured Formated Statement Report

LEVEL	COUNT		LEVEL	COUNT	
1	1	fich_inscr_ENT			<ENTITY>
2	2		2	2	etat_fich_inscr <ELEMENT> CONTAINED IN fich_inscr_ENT
3	2		2	2	support <ELEMENT> CONTAINED IN fich_inscr_ENT

LEVEL	COUNT		LEVEL	COUNT	
1	1	fich_signal_ENT			<ENTITY>

LEVEL	COUNT		LEVEL	COUNT	
1	1	identite_etd_ENT			<ENTITY>
2	2		2	2	etat_identite_etd <ELEMENT> CONTAINED IN identite_etd_ENT

LEVEL	COUNT		LEVEL	COUNT	
1	1	identite_txt_ENT			<ENTITY>
2	2		2	2	format_identite_txt <ELEMENT> CONTAINED IN identite_txt_ENT
3	2		2	2	contenu_identite_txt <ELEMENT> CONTAINED IN identite_txt_ENT

Interactive Design Approach IDA2.0R0 Page 6
FNDP - Namur VAX/VMS

Structured Formated Statement Report

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2
  1   1 lettre_accep_ENT          <ENTITY>
```

```
LEVEL COUNT
    1     1

  1   1 lettre_cont_ENT           <ENTITY>
  2    2  type_lettre_cont        <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  3    2  explication_lettre_cont <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  4    2  contenu                 <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  5    2  etat_lettre_cont        <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  6    2  echeance                <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  7    2  statut                  <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
  8    2  support                 <ELEMENT>
                                   CONTAINED IN lettre_cont_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     7

  1   1 nom_txt_ENT               <ENTITY>
  2    2  format_nom_txt          <ELEMENT>
                                        CONTAINED IN nom_txt_ENT
  3    2  contenu_nom_txt         <ELEMENT>
                                        CONTAINED IN nom_txt_ENT
```

Structured Formated Statement Report

LEVEL	COUNT		LEVEL	COUNT	
1	1	nom_var_ENT			<ENTITY>
2	2	format_nom_var			<ELEMENT> CONTAINED IN nom_var_ENT
3	2	val_def_nom_var			<ELEMENT> CONTAINED IN nom_var_ENT

LEVEL	COUNT		LEVEL	COUNT	
1	1	papiers_dus_ENT			<ENTITY>
2	2	etat_papiers_dus			<ELEMENT> CONTAINED IN papiers_dus_ENT
3	2	support			<ELEMENT> CONTAINED IN papiers_dus_ENT

LEVEL	COUNT		LEVEL	COUNT	
1	1	photo_ENT			<ENTITY>
2	2	type_photo			<ELEMENT> CONTAINED IN photo_ENT
3	2	explication_photo			<ELEMENT> CONTAINED IN photo_ENT
4	2	contenu			<ELEMENT> CONTAINED IN photo_ENT
5	2	etat_photo			<ELEMENT> CONTAINED IN photo_ENT
6	2	echeance			<ELEMENT> CONTAINED IN photo_ENT
7	2	statut			<ELEMENT> CONTAINED IN photo_ENT
8	2	support			<ELEMENT> CONTAINED IN photo_ENT

Structured Formated Statement Report

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     7


  1   1 prenom_txt_ENT              <ENTITY>
  2    2  format_prenom_txt         <ELEMENT>
                                      CONTAINED IN prenom_txt_ENT
  3    2  contenu_prenom_txt        <ELEMENT>
                                      CONTAINED IN prenom_txt_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1   1 prenom_var_ENT              <ENTITY>
  2    2  format_prenom_var         <ELEMENT>
                                      CONTAINED IN prenom_var_ENT
  3    2  val_def_prenom_var        <ELEMENT>
                                      CONTAINED IN prenom_var_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     2


  1   1 remplisseur_ENT             <ENTITY>
  2    2  format_remplisseur        <ELEMENT>
                                     CONTAINED IN remplisseur_ENT
  3    2  contenu_remplisseur       <ELEMENT>
                                     CONTAINED IN remplisseur_ENT
```

Structured Formated Statement Report

LEVEL	COUNT	LEVEL	COUNT
1	1	2	2

1	1	rentree_ENT	<ENTITY>
2	2	type_rentree	<ELEMENT> CONTAINED IN rentree_ENT
3	2	explication_rentree	<ELEMENT> CONTAINED IN rentree_ENT
4	2	contenu	<ELEMENT> CONTAINED IN rentree_ENT
5	2	etat_rentree	<ELEMENT> CONTAINED IN rentree_ENT
6	2	echeance	<ELEMENT> CONTAINED IN rentree_ENT
7	2	statut	<ELEMENT> CONTAINED IN rentree_ENT
8	2	support	<ELEMENT> CONTAINED IN rentree_ENT

LEVEL	COUNT	LEVEL	COUNT
1	1	2	7

1	1	reponse_ENT	<ENTITY>
2	2	identifiant_reponse	<ELEMENT> CONTAINED IN reponse_ENT
3	2	etat_reponse	<ELEMENT> CONTAINED IN reponse_ENT
4	2	echeance	<ELEMENT> CONTAINED IN reponse_ENT
5	2	support	<ELEMENT> CONTAINED IN reponse_ENT

LEVEL	COUNT	LEVEL	COUNT
1	1	2	4

1	1	secret_adm_ENT	<ENTITY>

Structured Formated Statement Report

```
LEVEL COUNT
    1     1


  1   1 secret_centr_ENT            <ENTITY>
```

```
LEVEL COUNT
    1     1


  1   1 tel_contact_ENT             <ENTITY>
  2    2  type_tel_contact          <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  3    2  explication_tel_contact   <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  4    2  contenu                   <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  5    2  etat_tel_contact          <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  6    2  echeance                  <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  7    2  statut                    <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
  8    2  support                   <ELEMENT>
                                      CONTAINED IN tel_contact_ENT
```

```
LEVEL COUNT   LEVEL COUNT
    1     1       2     7
```

B.5. Le rapport SELECTIVE-FORMATTED-STATEMENT

Il présente sous forme de phrases DSL, et pour une liste d'objets donnée par l'utilisateur, un sous-ensemble des informations stockées dans la base.

Exemple: fichier des objets reliés à d'autres par une clause "DOMAIN OF VALUE"

Commande: "SFS OPL FILE=OBJETS.DTA STAT=DOM.STT"

```
DEFINE ELEMENT                          contenu;

DEFINE ELEMENT                          contenu_annee_txt;
     DOMAIN OF VALUE ARE 'annee.';

DEFINE ELEMENT                          contenu_entete_txt;
     DOMAIN OF VALUE ARE 'Demande d''inscription';

DEFINE ELEMENT                          contenu_etd_sup_txt;
     DOMAIN OF VALUE ARE 'etudes superieures:';

DEFINE ELEMENT                          contenu_etudes_txt;
     DOMAIN OF VALUE ARE 'etudes.';

DEFINE ELEMENT                          contenu_identite_txt;
     DOMAIN OF VALUE ARE 'identite:';

DEFINE ELEMENT                          contenu_nom_txt;
     DOMAIN OF VALUE ARE 'nom:';

DEFINE ELEMENT                          contenu_prenom_txt;
     DOMAIN OF VALUE ARE 'prenom:';

DEFINE ELEMENT                          contenu_remplisseur;
     DOMAIN OF VALUE ARE 'a remplir par l''etudiant:';

DEFINE ELEMENT                          corps;

DEFINE ELEMENT                          date_exp_dde_inscr;

DEFINE ELEMENT                          date_exp_lettre_cont;
```

```
DEFINE ELEMENT                          date_exp_photo;

DEFINE ELEMENT                          date_exp_rentree;

DEFINE ELEMENT                          date_exp_reponse;

DEFINE ELEMENT                          date_exp_tel_contact;

DEFINE ELEMENT                          echeance;

DEFINE ELEMENT                          emplacement_annee_txt;
     DOMAIN OF VALUE ARE '2_1';

DEFINE ELEMENT                          emplacement_annee_var;
     DOMAIN OF VALUE ARE '3_1';
     DOMAIN OF VALUE ARE '4_1';
     DOMAIN OF VALUE ARE '5_1';

DEFINE ELEMENT                          emplacement_entete;
     DOMAIN OF VALUE ARE '1_1';

DEFINE ELEMENT                          emplacement_entete_txt;
     DOMAIN OF VALUE ARE '1_1';

DEFINE ELEMENT                          emplacement_etd_sup_txt;
     DOMAIN OF VALUE ARE '1_1';

DEFINE ELEMENT                          emplacement_etudes_sup;
     DOMAIN OF VALUE ARE '7_1';

DEFINE ELEMENT                          emplacement_etudes_txt;
     DOMAIN OF VALUE ARE '2_20';

DEFINE ELEMENT                          emplacement_etudes_var;
     DOMAIN OF VALUE ARE '3_20';
     DOMAIN OF VALUE ARE '4_20';
     DOMAIN OF VALUE ARE '5_20';


DEFINE ELEMENT                          emplacement_identite_etd;
     DOMAIN OF VALUE ARE '2_1';
```

```
DEFINE ELEMENT                          emplacement_identite_txt;
     DOMAIN OF VALUE ARE '2_1';

DEFINE ELEMENT                          emplacement_nom_txt;
     DOMAIN OF VALUE ARE '3_1';

DEFINE ELEMENT                          emplacement_nom_var;
     DOMAIN OF VALUE ARE '3_5';

DEFINE ELEMENT                          emplacement_prenom_txt;
     DOMAIN OF VALUE ARE '4_1';

DEFINE ELEMENT                          emplacement_prenom_var;
     DOMAIN OF VALUE ARE '4_8';

DEFINE ELEMENT                          emplacement_remplisseur;
     DOMAIN OF VALUE ARE '1_1';

DEFINE ELEMENT                          etat_dde_inscr;
     DOMAIN OF VALUE ARE a_remplir;
     DOMAIN OF VALUE ARE rempli;
     DOMAIN OF VALUE ARE a_corriger;
     DOMAIN OF VALUE ARE a_completer;
     DOMAIN OF VALUE ARE complet;
     DOMAIN OF VALUE ARE classe_inscr;
     DOMAIN OF VALUE ARE classe_etd_t;

DEFINE ELEMENT                          etat_doss_etd;
     DOMAIN OF VALUE ARE cree;
     DOMAIN OF VALUE ARE complet_t;

DEFINE ELEMENT                          etat_entete;
     DOMAIN OF VALUE ARE complet_t;

DEFINE ELEMENT                          etat_etudes_sup;
     DOMAIN OF VALUE ARE a_remplir;
     DOMAIN OF VALUE ARE rempli;
     DOMAIN OF VALUE ARE a_corriger;
     DOMAIN OF VALUE ARE a_completer;
     DOMAIN OF VALUE ARE complet_t;
```

```
DEFINE ELEMENT                          etat_fich_contact;
     DOMAIN OF VALUE ARE ouvert;
     DOMAIN OF VALUE ARE vide_t;


DEFINE ELEMENT                          etat_fich_inscr;
     DOMAIN OF VALUE ARE ouvert;
     DOMAIN OF VALUE ARE cloture;
     DOMAIN OF VALUE ARE vide_t;


DEFINE ELEMENT                          etat_identite_etd;
     DOMAIN OF VALUE ARE a_remplir;
     DOMAIN OF VALUE ARE rempli;
     DOMAIN OF VALUE ARE a_completer;
     DOMAIN OF VALUE ARE a_corriger;
     DOMAIN OF VALUE ARE complet_t;


DEFINE ELEMENT                          etat_lettre_cont;
     DOMAIN OF VALUE ARE recu_secr;
     DOMAIN OF VALUE ARE classe_cont;
     DOMAIN OF VALUE ARE classe_etd_t;


DEFINE ELEMENT                          etat_papiers_dus;
     DOMAIN OF VALUE ARE non_vide;
     DOMAIN OF VALUE ARE vide_t;


DEFINE ELEMENT                          etat_photo;
     DOMAIN OF VALUE ARE classe_dus;
     DOMAIN OF VALUE ARE classe_etd_t;


DEFINE ELEMENT                          etat_rentree;
     DOMAIN OF VALUE ARE en_attente;
     DOMAIN OF VALUE ARE survenu_t;


DEFINE ELEMENT                          etat_reponse;
     DOMAIN OF VALUE ARE envoye_etd_t;


DEFINE ELEMENT                          etat_tel_contact;
     DOMAIN OF VALUE ARE envoye_secr;
     DOMAIN OF VALUE ARE traite_t;


DEFINE ELEMENT                          explication_lettre_cont;


DEFINE ELEMENT                          explication_photo;
     DOMAIN OF VALUE ARE photographie;
```

```
DEFINE ELEMENT                          explication_rentree;

DEFINE ELEMENT                          explication_tel_contact;

DEFINE ELEMENT                          format_annee_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_annee_var;
     DOMAIN OF VALUE ARE 'aaaa/aaaa';

DEFINE ELEMENT                          format_entete_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_etd_sup_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_etudes_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_etudes_var;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_identite_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_nom_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_nom_var;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_prenom_txt;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_prenom_var;
     DOMAIN OF VALUE ARE 'alphabetique';

DEFINE ELEMENT                          format_remplisseur;
     DOMAIN OF VALUE ARE 'alphabetique';
```

```
DEFINE ELEMENT                          identifiant_dde_inscr;


DEFINE ELEMENT                          identifiant_doss_etd;

DEFINE ELEMENT                          identifiant_reponse;

DEFINE ELEMENT                          mot_cle;

DEFINE ELEMENT                          paragraphes_statiques;

DEFINE ELEMENT                          statut;
     DOMAIN OF VALUE ARE original;
     DOMAIN OF VALUE ARE copie;

DEFINE ELEMENT                          support;

DEFINE ELEMENT                          type_dde_inscr;
     DOMAIN OF VALUE ARE formulaire;

DEFINE ELEMENT                          type_lettre_cont;
     DOMAIN OF VALUE ARE lettre;

DEFINE ELEMENT                          type_photo;
     DOMAIN OF VALUE ARE autre;

DEFINE ELEMENT                          type_rentree;
     DOMAIN OF VALUE ARE calendrier;

DEFINE ELEMENT                          type_tel_contact;
     DOMAIN OF VALUE ARE telephone;

DEFINE ELEMENT                          val_def_annee_var;
     DOMAIN OF VALUE ARE '____/____';

DEFINE ELEMENT                          val_def_etudes_var;
     DOMAIN OF VALUE ARE '_____________________';

DEFINE ELEMENT                          val_def_nom_var;
     DOMAIN OF VALUE ARE '_____________________';
```

C. Le système d'interrogation de la base de données.

Il donne la possibilité de rechercher dans la base de données des objets qui vérifient des critères de sélection.

Exemples:

C.1. fichier des objets décrits

Sélection des noms des objets décrits:

```
CRITERION C1 = ENTITY;
SET S1 = C1;
```

C.2. fichier des expéditeurs

Sélection des relations traduisant l'expédition d'un objet:

```
CRITERION C2 = RELATION AND BN=?EXP?;
SET S2 = C2;
```

C.3. fichier des parties d'un formulaire

Sélection des relations traduisant la décomposition d'un formulaire en parties:

```
CRITERION C3 = RELATION AND BN=?COMP?;
SET S3 = C3;
```

3 LE PROGRAMME "CONSULTATION"

3.1 INTRODUCTION

Le programme "CONSULTATION" permet la transformation de la description des objets de bureau de DSL dans le MIB. Dans ce programme, il s'agira d'analyser les fichiers que peut fournir DSL et où l'on peut trouver l'information permettant le retour de DSL vers le MIB.

3.2 LES PRINCIPES DU PROGRAMME

Le programme "CONSULTATION" se compose de deux phases:

- la production des fichiers qui vont servir de base à la transformation des objets;
- l'analyse des fichiers pour retrouver l'information qui avait été décrite à partir du MIB.

3.2.1 LES FICHIERS NECESSAIRES A LA TRANSFORMATION

On sait quelle information on doit restituer à l'utilisateur. On dispose de commandes qui nous permettent d'extraire de l'information de la base de données des spécifications en DSL. Voyons comment utiliser ces commandes.

De quelle information a-t-on besoin ?

- L'utilisateur donne le nom de l'objet qu'il désire consulter.

- Recherche du type du message, du document et du formulaire, et de l'explication de ce type:
  On les a traduit comme étant les valeurs d'une clause "DOMAIN OF VALUE".

- Recherche des états d'un objet:
  On les a traduit comme étant les valeurs d'une clause "DOMAIN OF VALUE".

- Recherche des expéditeurs et des destinataires:
  Ils sont reliés à l'objet par une relation.

- Recherche des réponses attendues:
  Elles sont reliées à l'objet par une relation.

- Recherche des parties, éléments de texte et éléments variables d'un formulaire:
  Les composants d'un formulaire sont reliés à l'objet qu'ils composent par une relation.

- Recherche des format, texte et valeur par défaut des éléments de texte et éléments variables d'un formulaire:
  On les a traduit comme étant les valeurs d'une clause "DOMAIN OF VALUE".

- Recherche des constituants d'un objet structurant:
  Ils sont reliés à l'objet structurant par une relation.

La recherche des valeurs de la clause "DOMAIN OF VALUE" pourra se faire en analysant le fichier des objets reliés à d'autres par cette clause. Ce fichier peut être obtenu par la commande de production de rapport "SELECTIVE FORMATTED STATEMENT".

On pourra retrouver les relations entre les objets en utilisant le QUERY SYSTEM de DSL-SPEC: on recherchera les relations dont le nom de base contient certaines lettres marquant le type de la relation:

- les expéditeurs: EXP
- les destinataires: DEST
- les réponses attendues: REP
- les parties d'un formulaire: COMP
- les éléments de texte d'une partie: COMT
- les éléments variables d'une partie: COMV
- les correspondants des éléments variables: COR
- les constituants d'un objet structurant: CONST

Toutes les commandes de production de ces fichiers seront décrites dans le fichier "COMMANDES.DSA" dont le contenu est le suivant:

```
*
*
*  évocation du Query System
*
*
QS
*
*  Recherche du nom de tous les objets décrits.
*
CRITERION C0 = ENTITY;
SET ENS_ENT = C0;
PUNCH ENS_ENT ON OBJETS.DTA;
*
*  Recherche du nom de tous les attributs des objets décrits.
*
CRITERION C1 = ELEMENT;
SET ENS_ELT = C1;
PUNCH ENS_ELT ON ELEMENTS.DTA;
*
*  Recherche des expéditeurs des objets décrits.
*
CRITERION C2 = RELATION AND BN=?EXP?;
SET ENS_EXP = C2;
PUNCH ENS_EXP ON EXPEDITEURS.DAT;
*
*  Recherche des destinataires des objets décrits.
*
CRITERION C3 = RELATION AND BN=?DEST?;
SET ENS_DEST = C3;
PUNCH ENS_DEST ON DESTINATAIRES.DAT;
*
*  Recherche des objets attendus en réponse.
*
CRITERION C4 = RELATION AND BN=?REP?;
SET ENS_REP = C4;
PUNCH ENS_REP ON REPONSES.DAT;
*
*  Recherche des constituants d'un élément structurant.
*
CRITERION C5 = RELATION AND BN=?CONST?;
SET ENS_CONST = C5;
PUNCH ENS_CONST ON CONSTITUANTS.DAT;
*
*
```

```
*
*  Recherche des composants d'un formulaire.
*
CRITERION C6 = RELATION AND BN=?COR?;
SET ENS_CORR = C6;
PUNCH ENS_CORR ON CORRESPONDANTS.DAT;
*
CRITERION C7 = RELATION AND BN=?COMP?;
SET ENS_PART = C7;
PUNCH ENS_PART ON PARTIES .DAT;
*
CRITERION C8 = RELATION AND BN=?COMT?;
SET ENS_TXT = C8;
PUNCH ENS_TXT ON SQUELETTES.DAT;
*
CRITERION C9 = RELATION AND BN=?COMV?;
SET ENS_VAR = C9;
PUNCH ENS_VAR ON VARIABLES.DAT;
*
*  Fin d'utilisation du Query System
*
FINISH
*
*
*  Effectuer un "SELECTIVE FORMATTED STATEMENT"  pour trouver les
*  clauses "DOMAIN OF VALUE" des  éléments  sélectionnés  dans la
*  Query précédente au moyen du critère C1.
*
*
SFS OPL NP F=ELEMENTS.DTA NDLCC PUNCH=DOMAINES.DAT STAT=DOM.STT
*
*
*  Fin du fichier de commandes DSA
*
*
```

3.2.2 L'ANALYSE DES FICHIERS

Une fois que l'on dispose des fichiers extraits de la base de données des spécifications en DSL, il faudra les analyser afin de garnir des enregistrements avec l'information nécessaire pour le passage de DSL vers le MIB. Lorsqu'on aura complété ces enregistrements, on utilisera le gestionnaire d'écran TDMS pour afficher l'information au terminal.

3.3 LA DESCRIPTION DU PROGRAMME

3.3.1 LA GESTION D'ECRAN AU MOYEN DE TDMS

La gestion d'écran a été réalisée au moyen du gestionnaire d'écran TDMS.

A. LA DEFINITION DES ECRANS

Il s'agit ici de décrire les écrans tels qu'ils apparaissent à l'utilisateur. Les écrans se présentant de la même façon que dans le programme "BUREAU", on n'en donnera plus une description graphique mais une courte description en français.

a. l'écran "CONSULT_INITIAL_FORM"

Cet écran donne à l'utilisateur une description générale du programme "CONSULTATION".
Il se présente comme suit:

CECI EST LE PROGRAMME

CONSULTATION.

Ce programme permet de consulter les objets de bureau décrits en utilisant le programme "BUREAU", c'est-à-dire:
- les objets élémentaires;
- les objets structurants;

ainsi que les relations
- entre ces objets;
- entre ces objets et l'organisation.

Dans le menu principal, il vous suffira de donner le numéro et le nom de l'objet que vous désirez consulter.

b. l'écran "CONSULT_MENU_FORM"

Cet écran permet d'afficher le menu du programme. L'utilisateur doit donner le chiffre (donnée numérique) correspondant à son choix et le nom de l'objet (donnée alphanumérique) qu'il désire consulter. Une zone est également réservée pour afficher des messages d'erreur.

c. l'écran "CONSULT_MESSAGE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un mesage. Cet écran se présente de la même façon que l'écran "AJOUT_MESSAGE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

d. l'écran "CONSULT_DOCUMENT_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un document. Cet écran se présente de la même façon que l'écran "AJOUT_DOCUMENT_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

e. l'écran "CONSULT_FORMULAIRE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un formulaire. Cet écran se présente de la même façon que l'écran "AJOUT_FORMULAIRE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

f. l'écran "CONSULT_PARTIE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur une partie d'un formulaire. Cet écran se présente de la même façon que l'écran "AJOUT_PARTIE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

g. l'écran "CONSULT_TEXTE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un élément du squelette textuel d'un formulaire. Cet écran se présente de la même façon que l'écran "AJOUT_TEXTE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

h. l'écran "CONSULT_VARIABLE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un élément variable d'un formulaire. Cet écran se présente de la même façon que l'écran "AJOUT_VARIABLE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

i. l'écran "CONSULT_DOSSIER_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un dossier. Cet écran se présente de la même façon que l'écran "AJOUT_DOSSIER_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

j. l'écran "CONSULT_FICHIER_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur un fichier. Cet écran se présente de la même façon que l'écran "AJOUT_FICHIER_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

k. l'écran "CONSULT_PILE_FORM"

Cet écran permet à l'utilisateur de consulter les informations sur une pile. Cet écran se présente de la même façon que l'écran "AJOUT_PILE_FORM". Cependant, cette fois, toutes les données sont affichées et non collectées auprès de l'utilisateur.

B. LA DEFINITION DES ENREGISTREMENTS

Le programme "CONSULTATION" utilise les mêmes enregistrements que le programme "BUREAU". C'est pourquoi on ne redécrira pas ces enregistrements ici.

C. LES REQUETES

a. la requête "CONSULT_INITIAL_REQUEST"

Cette requête permet d'afficher l'écran "CONSULT_INITIAL_FORM".

Description de la requête:

```
REPLACE REQUEST CONSULT_INITIAL_REQUEST;

DESCRIPTION  /*  Cette requête affiche un texte descrptif.
                 Elle explique comment utiliser le programme
                 "CONSULTATION"                              */;

FORM IS CONSULT_INITIAL_FORM;

CLEAR SCREEN;
DISPLAY FORM CONSULT_INITIAL_FORM;

MESSAGE LINE IS "Tapez RETURN pour afficher le menu.";

WAIT;

END DEFINITION;
```

b. la requête "CONSULT_MENU_REQUEST"

Cette requête permet d'afficher l'écran "CONSULT_MENU_FORM". Elle range dans l'enregistrement "BUREAU_TRAVAIL_RECORD" la sélection de l'utilisateur et le nom de l'objet qu'il désire consulter.

Description de la requête:

```
REPLACE REQUEST CONSULT_MENU_REQUEST;

DESCRIPTION  /*  Cette requête affiche un menu des choix de
                 l'application.
                 Elle collecte un numéro de sélection et le nom
                 de l'objet à ajouter.                       */;

FORM IS CONSULT_MENU_FORM;

RECORD IS BUREAU_TRAVAIL_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_MENU_FORM;

   OUTPUT MESS_ERREUR TO MESS_ERREUR;
   INPUT SELECTION TO SELECTION;
   INPUT CONSULT_OBJET TO CONSULT_OBJET;

END DEFINITION;
```

c. la requête "CONSULT_MESSAGE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_MESSAGE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_MESSAGE_REQUEST;

DESCRIPTION  /*  Affiche les informations  concernant  un message
                 dont on a donné le nom.                    */;

FORM IS CONSULT_MESSAGE_FORM;

RECORD IS MESSAGE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_MESSAGE_FORM;

   OUTPUT NOM_MESS TO NOM_MESS;
   OUTPUT TYPE_MESS TO TYPE_MESS;
   OUTPUT EXPLICATION_MESS TO EXPLICATION_MESS;
   OUTPUT ETATS_MESS[1 TO 10] TO ETATS_MESS[1 TO 10];
   OUTPUT EXPED_MESS TO EXPED_MESS;
   OUTPUT DEST_MESS[1 TO 10] TO DEST_MESS[1 TO 10];
   OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];

   INPUT DUM_ETAT[1 TO 10] TO DUM_ETAT[1 TO 10];
   INPUT DUM_DEST[1 TO 10] TO DUM_DEST[1 TO 10];
   INPUT DUM_REP[1 TO 10] TO DUM_REP[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

d. la requête "CONSULT_DOCUMENT_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_DOCUMENT_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_DOCUMENT_REQUEST;

DESCRIPTION  /*  Affiche les informations  concernant un document
                 dont on a donné le nom.                    */;

FORM IS CONSULT_DOCUMENT_FORM;

RECORD IS DOCUMENT_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_DOCUMENT_FORM;

   OUTPUT NOM_DOC TO NOM_DOC;
   OUTPUT TYPE_DOC TO TYPE_DOC;
   OUTPUT EXPLICATION_DOC TO EXPLICATION_DOC;
   OUTPUT ETATS_DOC[1 TO 10] TO ETATS_DOC[1 TO 10];
   OUTPUT EXPED_DOC TO EXPED_DOC;
   OUTPUT DEST_DOC[1 TO 10] TO DEST_DOC[1 TO 10];
   OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];

   INPUT DUM_ETAT[1 TO 10] TO DUM_ETAT[1 TO 10];
   INPUT DUM_DEST[1 TO 10] TO DUM_DEST[1 TO 10];
   INPUT DUM_REP[1 TO 10] TO DUM_REP[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

e. la requête "CONSULT_FORMULAIRE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_FORMULAIRE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_FORMULAIRE_REQUEST;

DESCRIPTION /* Affiche les informations concernant un
               formulaire dont on a donné le nom.      */;

FORM IS CONSULT_FORMULAIRE_FORM;

RECORD IS FORMULAIRE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_FORMULAIRE_FORM;

   OUTPUT TITRE_FORM TO TITRE_FORM;
   OUTPUT TYPE_FORM TO TYPE_FORM;
   OUTPUT ETATS_FORM[1 TO 10] TO ETATS_FORM[1 TO 10];
   OUTPUT EXPED_FORM TO EXPED_FORM;
   OUTPUT DEST_FORM[1 TO 10] TO DEST_FORM[1 TO 10];
   OUTPUT NOM[1 TO 10] TO NOM[1 TO 10];

   INPUT DUM_ET_FORM[1 TO 10] TO DUM_ET_FORM[1 TO 10];
   INPUT DUM_DES_FORM[1 TO 10] TO DUM_DES_FORM[1 TO 10];
   INPUT DUM_REP[1 TO 10] TO DUM_REP[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

f. la requête "CONSULT_PARTIE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_PARTIE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_PARTIE_REQUEST;

DESCRIPTION  /*  Affiche les informations concernant une partie
                 d'un formulaire dont on a donné le nom.   */;

FORM IS CONSULT_PARTIE_FORM;

RECORD IS PARTIE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_PARTIE_FORM;

   OUTPUT NOM_PARTIE TO NOM_PARTIE;
   OUTPUT LIGNE_PART[1 TO 10] TO LIGNE_PART[1 TO 10];
   OUTPUT COLONNE_PART[1 TO 10] TO COLONNE_PART[1 TO 10];
   OUTPUT ETATS_PARTIE[1 TO 10] TO ETATS_PARTIE[1 TO 10];

   INPUT DUM_ET_EMPLAC[1 TO 10] TO DUM_ET_EMPLAC[1 TO 10];
   INPUT DUM_ET_PARTIE[1 TO 10] TO DUM_ET_PARTIE[1 TO 10];

WAIT;

END DEFINITION;
```

g. la requête "CONSULT_TEXTE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_TEXTE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_TEXTE_REQUEST;

DESCRIPTION /* Affiche les informations sur le squelette
               textuel d'un formulaire dont on a donné
               le nom. */;

FORM IS CONSULT_TEXTE_FORM;

RECORD IS TEXTE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_TEXTE_FORM;

   OUTPUT NOM_TEXTE TO NOM_TEXTE;
   OUTPUT FORMAT TO FORMAT;
   OUTPUT CONTENU TO CONTENU;
   OUTPUT LIGNE_TXT[1 TO 10] TO LIGNE_TXT[1 TO 10];
   OUTPUT COLONNE_TXT[1 TO 10] TO COLONNE_TXT[1 TO 10];

   INPUT DUM_TXT[1 TO 10] TO DUM_TXT[1 TO 10];

WAIT;

END DEFINITION;
```

h. la requête "CONSULT_VARIABLE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_VARIABLE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_VARIABLE_REQUEST;

DESCRIPTION /* Affiche les informations sur les elements
               variables d'un formulaire dont on a donné
               le nom.  */;

FORM IS CONSULT_VARIABLE_FORM;

RECORD IS VARIABLE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_VARIABLE_FORM;

   OUTPUT NOM_VAR TO NOM_VAR;
   OUTPUT FORMAT TO FORMAT;
   OUTPUT VAL_DEF TO VAL_DEF;
   OUTPUT LIGNE_VAR[1 TO 10] TO LIGNE_VAR[1 TO 10];
   OUTPUT COLONNE_VAR[1 TO 10] TO COLONNE_VAR[1 TO 10];
   OUTPUT TXT_COR TO TXT_COR;

   INPUT DUM_VAR[1 TO 10] TO DUM_VAR[1 TO 10];

WAIT;

END DEFINITION;
```

i. la requête "CONSULT_DOSSIER_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_DOSSIER_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_DOSSIER_REQUEST;

DESCRIPTION  /*  Affiche les informations  concernant  un dossier
                 dont on a donné le nom.                    */;

FORM IS CONSULT_DOSSIER_FORM;

RECORD IS DOSSIER_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_DOSSIER_FORM;

   OUTPUT NOM_DOSSIER TO NOM_DOSSIER;
   OUTPUT ETATS_DOSS[1 TO 10] TO ETATS_DOSS[1 TO 10];
   OUTPUT EXPED_DOSS TO EXPED_DOSS;
   OUTPUT DEST_DOSS[1 TO 10] TO DEST_DOSS[1 TO 10];
   OUTPUT CONST_DOSS[1 TO 20] TO CONST_DOSS[1 TO 20];

   INPUT DUM_ET_DOSS[1 TO 10] TO DUM_ET_DOSS[1 TO 10];
   INPUT DUM_DES_DOSS[1 TO 10] TO DUM_DES_DOSS[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

j. la requête "CONSULT_FICHIER_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_FICHIER_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_FICHIER_REQUEST;

DESCRIPTION /* Affiche les informations concernant un fichier
               dont on a donné le nom.                    */;

FORM IS CONSULT_FICHIER_FORM;

RECORD IS FICHIER RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_FICHIER_FORM;

   OUTPUT NOM_FICHIER TO NOM_FICHIER;
   OUTPUT ETATS_FICH[1 TO 10] TO ETATS_FICH[1 TO 10];
   OUTPUT CONST_FICH TO CONST_FICH;

   INPUT DUM_ET_FICH[1 TO 10] TO DUM_ET_FICH[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

k. la requête "CONSULT_PILE_REQUEST";

Cette requête permet d'afficher l'écran "CONSULT_PILE_FORM". Elle complète l'écran en affichant les renseignements qui avaient été préalablement décrits par l'utilisateur.

Description de la requête:

```
REPLACE REQUEST CONSULT_PILE_REQUEST;

DESCRIPTION /* Affiche les informations concernant une pile
               dont on a donné le nom.                    */;

FORM IS CONSULT_PILE_FORM;

RECORD IS PILE_RECORD;

CLEAR SCREEN;
DISPLAY FORM CONSULT_PILE_FORM;

   OUTPUT NOM_PILE TO NOM_PILE;
   OUTPUT ETATS_PILE[1 TO 10] TO ETATS_PILE[1 TO 10];

   INPUT DUM_ET_PILE[1 TO 10] TO DUM_ET_PILE[1 TO 10];

MESSAGE LINE IS "Tapez RETURN pour revenir au menu.";

WAIT;

END DEFINITION;
```

l. la construction de la librairie des requêtes

Toutes les requêtes décrites précédemment sont rangées dans le fichier de librairie des requêtes "CONSULIB.RLB".

```
REPLACE LIBRARY CONSULTATION_LIBRARY;
  REQUEST IS CONSULT_INITIAL_REQUEST;
  REQUEST IS CONSULT_MENU_REQUEST;
  REQUEST IS CONSULT_MESSAGE_REQUEST;
  REQUEST IS CONSULT_DOCUMENT_REQUEST;
  REQUEST IS CONSULT_FORMULAIRE_REQUEST;
  REQUEST IS CONSULT_PARTIE_REQUEST;
  REQUEST IS CONSULT_TEXTE_REQUEST;
  REQUEST IS CONSULT_VARIABLE_REQUEST;
  REQUEST IS CONSULT_DOSSIER_REQUEST;
  REQUEST IS CONSULT_FICHIER_REQUEST;
  REQUEST IS CONSULT_PILE_REQUEST;
  FILE IS "CONSULIB";
END DEFINITION;
```

3.3.2 LE PROGRAMME DE TRANSFORMATION EN C

On donnera d'abord la spécification du programme et de chaque fonction utilisée. On peut trouver ensuite le texte du programme dans le langage C.

A. SPECIFICATION

Pour chacune des fonctions, on donnera:
- une brève description en français de son effet;
- ses arguments;
- ses résultats;
- sa spécification sous forme de préconditions et postconditions;
- éventuellement, une description de son algorithme.

a. Le programme principal

* effet:

Ce programme a pour objet de présenter à l'utilisateur les informations concernant un objet dont il donne le nom et qu'il aura décrit préalablement dans le programme "BUREAU".

* arguments:
  - les fichiers contenant les descriptions partielles des objets en DSL:
    - OBJETS;
    - EXPEDITEURS;
    - DESTINATAIRES;
    - REPONSES;
    - CONSTITUANTS;
    - PARTIES;
    - SQUELETTES;
    - VARIABLES;
    - CORRESPONDANTS;
    - DOMAINES.

* pré:
  - tous les fichiers décrits ci_dessus existent et ne sont pas vides

* post:
  - le programme affiche à l'écran les différents objets que l'utilisateur désire consulter

* algorithme:
  - ouverture de tous les fichiers de données
    et de tous les fichiers nécessaires à la gestion d'écran au moyen de TDMS
  - affichage de l'explication du programme
  - tant que l'utilisateur désire consulter des objets:
    - affichage du menu de sélection
    - appel de la fonction de consultation correspondant à la sélection
  - fermeture de tous les fichiers

b. La fonction CCOPY

* effet:
  Cette fonction ajoute à une chaine de caractères fournie par TDMS le caractère "\0" qui marque la fin d'une chaine de caractères en C.

* arguments:
  - origne: chaine fournie par TDMS
  - bmax: borne supérieure maximum de cette chaine

* résultats:
  - copie: chaine traduite en C

* pré:
  - il existe une chaine "origine"

* post:
  - copie = origine + "\0"

* algorithme:
  - en commençant par la fin de la chaine offerte par TDMS et tant qu'on a à faire à un caractère blanc, on passe ce caractère.
  - on ajoute le caractère "0" à la fin de la chaine.

c. La fonction TDMSCOPY

* effet:
Cette fonction enlève d'une chaine de caractères fournie par C le caractère "\0" qui en marque la fin et elle termine cette chaine par des caractères blancs.

* arguments:
- original: chaine fournie par C
- bmax: borne supérieure maximum de cette chaine

* résultats:
- copie: chaine traduite en TDMS

* pré:
- il existe une chaine "origine"

* post:
- copie = origine - "\0"

d. La fonction TEST

* effet: Cette fonction teste le Return Status renvoyé par les fonctions de TDMS.

* arguments:
- RET_STAT: Return Status

* pré:
- il existe un Return Status RET_STAT

* post:
- si RET_STAT <> valeur normale, le programme est arrêté

e. La fonction EXIST

* effet:
Cette fonction teste si l'objet que l'on désire consulter existe.

* arguments:
- objet: nom de l'objet que l'on désire consulter

* résultats:
- TROUVE: témoin de l'existence de l'objet

* pré:
- il existe un nom d'objet "objet"

* post:
- si "objet" existe, alors TROUVE = vrai
sinon TROUVE = faux

f. La fonction ASSOC1

* effet:
Cette fonction permet de trouver un objet relié à un autre par une seule relation d'un certain type. Elle servira à trouver l'expéditeur d'un objet, le constituant d'un fichier ou le correspondant d'un élément variable d'un formulaire.

* arguments:
- objet: objet que l'on désire consulter
- relation: abréviation de la relation étudiée
- fichier: fichier dans lequel on lit la relation

* résultats:
- associé: objet associé recherché

* pré:
- l'objet "objet" existe et l'abréviation de la relation a la valeur "_EXP_" ou "_CONS_" ou "_COR_"

* post:
- "associé" contient l'objet associé à "objet" par la relation du type "relation"

g. La fonction ASSOC2

* effet:
Cette fonction permet de trouver les objets reliés à un autre par une relation d'un certain type. Elle servira à retrouver les composants (partie, élément de texte et élément variable) d'un formulaire; elle servira aussi à retrouver les destinataires et les réponses d'un objet et les constituants d'un dossier.

* arguments:
- objet: objet consulté
- relation: abréviation de la relation étudiée
- fichier: fichier dans lequel on lit les associations
- nbre_comp: nombre maximum d'objets reliés à l'objet étudié

* résultats:
- composant: tableau comprenant les objets reliés à l'objet consulté

* pré:
- l'objet "objet" existe et l'abréviation de la relation a la valeur "_COMP_" ou "_COMT_" ou "_COMV_" ou "_DEST_" ou "_REP_" ou "_CONS_"

* post:
- "composant" contient les objets associés à "objet" par la relation de type "relation"

h. La fonction PRESENT

* effet:
Cette fonction permet de voir si une chaine de caractères en contient une autre plus petite.

* arguments:
- grand: grande chaine de caractères
- petit: petite chaine de caractères

* résultats:
- TROUVE: témoin de la présence de la petite chaine dans la grande

* pré:
- longueur de la chaine "grand" >= longueur de la chaine "petit"

* post:
- TROUVE = vrai si la chaine "petit" est présente dans la chaine "grand"
TROUVE = faux sinon

i. La fonction DOM1

* effet:
Cette fonction permet de retrouver les valeurs des attributs "type" du message, du document et du formulaire et "explication" du message et du document.

* arguments:
- objet: objet consulté
- type: type de l'attribut étudié
- fichier: fichier où on va lire l'attribut

* résultats:
- attribut: attribut recherché

* pré:
- l'objet "objet" existe et le type de l'attribut a les valeurs "type_" ou "explication_"

* post:
- "attribut" contient la valeur de l'attribut recherché de l'objet "objet"

j. La fonction DOM2

* effet:
Cette fonction permet de retrouver les valeurs des différents états des objets de bureau.

* arguments:
- objet: objet consulté
- type: type de l'attribut étudié
- fichier: fichier où on lit les attributs

* résultats:
- attribut: tableau contenant les attributs

* pré:
- l'objet "objet" existe et le type de l'attribut a la valeur "etat_"

* post:
- le tableau "attribut" contient les valeurs des attributs du type "type" de l'objet "objet"

k. La fonction DOM3

* effet:
Cette fonction permet de retrouver les valeurs des attributs "format" des éléments du squelette textuel et des éléments variables, "contenu" des éléments de texte et "val_def" des éléments variables d'un formulaire.

* arguments:
- objet: objet consulté
- type: type de l'attribut étudié
- fichier: fichier où on lit les attributs

* résultats:
- attribut: attribut recherché

* pré:
- l'objet existe et le type de l'attribut a la valeur "format_" ou "contenu_" ou "val_def_"

* post:
- "attribut" contient la valeur de l'attribut de type "type" de l'objet "objet"

l. La fonction DOM4

* effet:
Cette fonction permet de retrouver les valeurs de l'attribut "emplacement" des différentes parties, éléments du squelette textuel et éléments variables d'un formulaire.

* arguments:
- objet: objet consulté
- type: type de l'attribut étudié
- fichier: fichier où on lit les emplacements

* résultats:
- tabligne: tableau des numéros de ligne
- tabcolonne: tableau des numéros de colonne

* pré:
- l'objet existe et le type de l'attribut a la valeur "emplacement_"

* post:
- les tableaux "tabligne" et "tabcolonne" contiennent les coordonnées de l'objet "objet"

m. La fonction CONS_MESSAGE

* effet:
Cette fonction recherche les informations concernant un message que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom du message que l'on désire consulter

* pré:
- le nom "nom_obj" du message existe

* post:
- le programme affiche à l'écran les renseignements concernant le message "nom_obj"

n. La fonction CONS_DOCUMENT

* effet:
Cette fonction recherche les informations concernant un document que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom du document que l'on désire consulter

* pré:
- le nom "nom_obj" du document existe

* post:
- le programme affiche à l'écran les renseignements concernant le document "nom_obj"

o. La fonction CONS_DOSSIER

* effet:
Cette fonction recherche les informations concernant un dossier que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom du dossier que l'on désire consulter

* pré:
- le nom "nom_obj" du dossier existe

* post:
- le programme affiche à l'écran les renseignements concernant le dossier "nom_obj"

p. La fonction CONS_FICHIER

* effet:
Cette fonction recherche les informations concernant un fichier que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom du fichier que l'on désire consulter

* pré:
- le nom "nom_obj" du fichier existe

* post:
- le programme affiche à l'écran les renseignements concernant le fichier "nom_obj"

q. La fonction CONS_PILE

* effet:
Cette fonction recherche les informations concernant une pile que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom de la pile que l'on désire consulter

* pré:
- le nom "nom_obj" de la pile existe

* post:
- le programme affiche à l'écran les renseignements concernant la pile "nom_obj"

r. La fonction CONS_FORMULAIRE

* effet:
Cette fonction recherche les informations concernant les généralités sur un formulaire que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_obj: nom du formulaire que l'on désire consulter

* pré:
- le nom "nom_obj" du formulaire existe

* post:
- le programme affiche à l'écran les renseignements généraux concernant le formulaire "nom_obj"

s. La fonction CONS_PARTIE

* effet:

Cette fonction recherche les informations concernant une partie d'un formulaire que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom_part: nom de la partie de formulaire que l'on désire consulter

* pré:
- le nom "nom_part" de la partie de formulaire existe

* post:
- le programme affiche à l'écran les renseignements concernant la partie de formulaire "nom_part"

t. La fonction CONS_TEXTE

* effet:
Cette fonction recherche les informations concernant un élément du squelette textuel d'un formulaire que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom: nom de l'élément du squelette textuel du formulaire que l'on désire consulter

* pré:
- le nom "nom_" de l'élément du squelette textuel du formulaire existe

* post:
- le programme affiche à l'écran les renseignements concernant l'élément "nom" du squelette textuel d'un formulaire

u. La fonction CONS_VARIABLE

* effet:
Cette fonction recherche les informations concernant un élément variable d'un formulaire que l'on désire consulter, à partir de sa traduction en DSL.

* arguments:
- nom: nom de l'élément variable du formulaire que l'on désire consulter

* pré:
- le nom "nom" de l'élément variable existe

* post:
- le programme affiche à l'écran les renseignements concernant l'élément variable "nom" d'un formulaire

B. TEXTE DU PROGRAMME

```
/*****************************************************************/
/*****************************************************************/
/**********     PROGRAMME  CONSULTATION                **********/
/*****************************************************************/
/*****************************************************************/
/**** Ce programme permet de consulter les objets de bureau ****/
/**** préalablement décrits à l'aide du programme BUREAU.   ****/
/*****************************************************************/
/*****************************************************************/

#include stdio      /* définitions relatives aux input/output */

#include descrip    /* définitions relatives aux appels par
                        descripteurs */


/* inclusion des enregistrements définis dans TDMS */

#dictionary "BUREAU_TRAVAIL_RECORD"
#dictionary "MESSAGE_RECORD"
#dictionary "DOCUMENT_RECORD"
#dictionary "FORMULAIRE_RECORD"
#dictionary "PARTIE_RECORD"
#dictionary "TEXTE_RECORD"
#dictionary "VARIABLE_RECORD"
#dictionary "DOSSIER_RECORD"
#dictionary "FICHIER_RECORD"
#dictionary "PILE_RECORD"


/* déclaration des constantes */

#define  TRUE  1
#define  FALSE 0


/* déclaration des variables */

int RET_STAT;        /* Return Status des fonctions de TDMS */
int CHANNEL;         /* canal vers le terminal */
int LIBRARY_ID;      /* identificateur de la librairie de TDMS */
int BON_SELEC;       /* témoin de bonne sélection */
```

```
/* déclaration des structures de saisie dans TDMS */

struct   bureau_travail_rec   trav_bur;
struct   message_rec          msge;
struct   document_rec         doct;
struct   formulaire_rec       form;
struct   partie_rec           part;
struct   texte_rec            elt_txt;
struct   variable_rec         elt_var;
struct   dossier_rec          doss;
struct   fichier_rec          fich;
struct   pile_rec             tas;


/* déclaration des fichiers */

FILE   *fopen( );
FILE   *OBJETS;
FILE   *EXPEDITEURS;
FILE   *DESTINATAIRES;
FILE   *REPONSES;
FILE   *CONSTITUANTS;
FILE   *PARTIES;
FILE   *SQUELETTES;
FILE   *VARIABLES;
FILE   *CORRESPONDANTS;
FILE   *DOMAINES;


/* déclaration des fonctions de TDMS */

extern  int  TSS$OPEN( );
extern  int  TSS$OPEN_RLB( );
extern  int  TSS$REQUEST( );
extern  int  TSS$CLOSE( );
extern  int  TSS$CLOSE_RLB( );
extern  int  LIB$STOP( );
```

```
/* déclaration des descripteurs de la librairie et
   des requêtes de TDMS */

$DESCRIPTOR (LIB_DESC,"CONSULIB.RLB");
$DESCRIPTOR (INIT_DESC,"CONSULT_INITIAL_REQUEST");
$DESCRIPTOR (MENU_DESC,"CONSULT_MENU_REQUEST");
$DESCRIPTOR (MESS_DESC,"CONSULT_MESSAGE_REQUEST");
$DESCRIPTOR (DOC_DESC,"CONSULT_DOCUMENT_REQUEST");
$DESCRIPTOR (DOSS_DESC,"CONSULT_DOSSIER_REQUEST");
$DESCRIPTOR (FICH_DESC,"CONSULT_FICHIER_REQUEST");
$DESCRIPTOR (PILE_DESC,"CONSULT_PILE_REQUEST");
$DESCRIPTOR (FORM_DESC,"CONSULT_FORMULAIRE_REQUEST");
$DESCRIPTOR (PART_DESC,"CONSULT_PARTIE_REQUEST");
$DESCRIPTOR (TEXTE_DESC,"CONSULT_TEXTE_REQUEST");
$DESCRIPTOR (VAR_DESC,"CONSULT_VARIABLE_REQUEST");


/* déclaration des structures utilisées en C */

struct  consult_trav
  {
    int  c_select;
    char c_mess_err[81];
    char c_consult_objet[13];
  }   trav_cons;
```

```
/*****************************************************************/
/*****************************************************************/
/****                                                         ****/
/****                  PROGRAMME  PRINCIPAL                   ****/
/****                                                         ****/
/*****************************************************************/
/*****************************************************************/


main( )
{
   /* ouverture des fichiers de données */

   if ((OBJETS = fopen("OBJETS.DTA","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((EXPEDITEURS = fopen("EXPEDITEURS.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((DESTINATAIRES = fopen("DESTINATAIRES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((REPONSES = fopen("REPONSES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((CONSTITUANTS = fopen("CONSTITUANTS.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((PARTIES = fopen("PARTIES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((SQUELETTES = fopen("SQUELETTES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((VARIABLES = fopen("VARIABLES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((CORRESPONDANTS = fopen("CORRESPONDANTS.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");

   if ((DOMAINES = fopen("DOMAINES.DAT","r")) == NULL)
        printf("erreur: fichier non ouvert\n");


   /* ouverture de la librairie des requêtes */

   RET_STAT = TSS$OPEN_RLB(&LIB_DESC,&LIBRARY_ID);
   test(RET_STAT);

   /* ouverture du canal vers le terminal */

   RET_STAT = TSS$OPEN(&CHANNEL);
   test(RET_STAT);
```

```
/* affichage de l'explication du programme */

RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&INIT_DESC);
test(RET_STAT);

/* consultation des objets de bureau */

trav_bur.selection = 0;
while (trav_bur.selection != 2)
      /* tant que je veux consulter des objets */
{
   strncpy(trav_bur.mess_erreur,"                ",80);
   BON_SELEC = FALSE;
   while (BON_SELEC == FALSE)
   {
      /* affichage de la selection */

      trav_bur.selection = 0;
      strncpy(trav_bur.consult_objet,"            ",12);
      RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&MENU_DESC,
                             &trav_bur);
      test(RET_STAT);

      /* traduction de la structure TDMS en une structure C */

      trav_cons.c_select = trav_bur.selection;
      ccopy(trav_cons.c_mess_err,trav_bur.mess_erreur,80);
      ccopy(trav_cons.c_consult_objet,
                       trav_bur.consult_objet,12);

      /* test des paramètres rentrés lors de la sélection */

      if (trav_cons.c_select != 2)
      {
         if ((exist(trav_cons.c_consult_objet)) == FALSE)
           strncpy(trav_bur.mess_erreur,
                   "L'objet a consulter n'existe pas",80);
         else  BON_SELEC = TRUE;
      }
      if (trav_cons.c_select == 2)
        BON_SELEC = TRUE;
   } /* fin du while BON_SELEC */


   /* selon la selection, appel aux fonctions de consultation
      d'information sur les objets traduits en DSL         */

   switch (trav_bur.selection)
   {
      case 11: cons_message(trav_cons.c_consult_objet);
               break;

      case 12: cons_document(trav_cons.c_consult_objet);
               break;
```

```
            case 13: cons_formulaire(trav_cons.c_consult_objet);
                     break;

            case 14: cons_dossier(trav_cons.c_consult_objet);
                     break;

            case 15: cons_fichier(trav_cons.c_consult_objet);
                     break;

            case 16: cons_pile(trav_cons.c_consult_objet);
                     break;
         }   /* fin du switch sur selection */

      }  /* fin du while sur trav_bur.selection  */


      /* refermer tous les fichiers */

      fclose(OBJETS);
      fclose(EXPEDITEURS);
      fclose(DESTINATAIRES);
      fclose(REPONSES);
      fclose(CONSTITUANTS);
      fclose(PARTIES);
      fclose(SQUELETTES);
      fclose(VARIABLES);
      fclose(CORRESPONDANTS);
      fclose(DOMAINES);
      RET_STAT = TSS$CLOSE_RLB(&LIBRARY_ID);
      test(RET_STAT);
      RET_STAT = TSS$CLOSE(&CHANNEL);
      test(RET_STAT);

}  /* fin du programme principal */




/*****************************************************************/
```

```
/*****************************************************************/
/****                                                         ****/
/****                      FONCTIONS                          ****/
/****                                                         ****/
/*****************************************************************/


/*****************************************************************/
/****************      FONCTION  CCOPY          ******************/
/*****************************************************************/


ccopy(copie,origine,bmax)
   char  *origine;
   char  *copie;
   int   bmax;

{
   int  i;

   i = bmax - 1;
   while ((origine[i] == ' ')
          && ( i >= 0 ))
   {
       i = i - 1;
   }
   copie[i +1] = '\0';
   while ( i >= 0)
   {
       copie[i] = origine[i];
       i = i - 1;
   }

}
```

```
/*****************************************************************/
/****************      FONCTION TDMSCOPY          ****************/
/*****************************************************************/


TDMScopy(original,copie,bmax)

   char *original;
   char *copie;
   int bmax;

{
   int i;

   i = 0;
   while (original[i] != '\0')
   {
      copie[i] = original[i];
      i = i + 1;
   }
   while (i < bmax)
   {
      copie[i] = ' ';
      i = i + 1;
   }
}



/*****************************************************************/
/****************       FONCTION  TEST            ****************/
/*****************************************************************/


test(RET_STAT)
{
   if (RET_STAT != 2981889)   /* si RET_STAT != TSS$NORMAL */
     LIB$STOP(RET_STAT);
}
```

```
/*****************************************************************/
/****************        FONCTION  EXIST          ****************/
/*****************************************************************/


int exist(objet)

   char  objet[13];

{
   int  TROUVE = FALSE;
   char buffer[18];
   int  i;

   fseek(OBJETS,0,0);
   while ((TROUVE == FALSE)
          && ( fgets(buffer,18,OBJETS) != NULL))
   {
       for ( i = 17 ; i >= 0 ; i--)
       {
           if (buffer[i] == 'E')
           {
               buffer[i - 1] = '\0';
               break;
           }
       }
       if (strcmp(buffer,objet) == 0)
               TROUVE = TRUE;
   }
   return(TROUVE);
}
```

```
/*****************************************************************/
/****************      FONCTION  ASSOC1          *****************/
/*****************************************************************/


assoc1(objet,associe,relation,fichier)

   char objet[13];     /* objet donné */
   char associe[13];   /* objet associé recherché */

   char relation[7];   /* abréviation de la relation étudiée */
   FILE *fichier;      /* fichier dans lequel on lit
                          l'association */

{
   char concatenation[19];
   char association[30];
   int TROUVE;
   int i,j;

   strcpy(concatenation,objet);
   strcat(concatenation,relation);
   fseek(fichier,0,0);
   TROUVE = FALSE;
   while ((TROUVE == FALSE) &&
         (fgets(association,30,fichier) != NULL))
   {
      TROUVE = TRUE;
      i = 0;
      while ((i < strlen(concatenation))
            && (TROUVE == TRUE))
      {
         if (association[i] != concatenation[i])
               TROUVE = FALSE;
         else i = i + 1;
      }
   }
   if (TROUVE == TRUE)
   {
      j = 0;
      i = strlen(concatenation);
      while ((association[i] != '\0')
            && (association[i] != 10))
      {
          associe[j] = association[i];
          i = i + 1;
          j = j + 1;
      }
      associe[j] = '\0';
   }
   else associe[0] = '\0';
}     /* fin de la fontion assoc1 */
```

```
/*****************************************************************/
/****************    FONCTION  ASSOC2          ****************/
/*****************************************************************/

assoc2(objet,composant,nbre_comp,relation,fichier)

   char objet[13];
   char composant[][13];
   int  nbre_comp;
   char relation[7];
   FILE *fichier;

{
   char concatenation[19];
   char association[30];
   int TROUVE;
   int i,j,k;

   for ( i = 0 ; i < nbre_comp ; i++ )
       composant[i][0] = '\0';
   strcpy(concatenation,objet);
   strcat(concatenation,relation);
   fseek(fichier,0,0);
   i = 0;
   while ((i < nbre_comp) &&
          (fgets(association,30,fichier) != NULL))
   {
      TROUVE = TRUE;
      j = 0;
      while (( j < strlen(concatenation)) &&
             (TROUVE == TRUE))
      {
         if (association[j] != concatenation[j])
                TROUVE = FALSE;
         else  j = j + 1;
      }
      if (TROUVE == TRUE)
      {
         k = 0;
         j = strlen(concatenation);
         while ((association[j] != '\0')
                && (association[j] != 10))
         {
            composant[i][k] = association[j];
            k = k + 1;
            j = j + 1;
         }
         composant[i][k] = '\0';
         i = i + 1;
      }
   }
}   /* fin de la fonction assoc2 */
```

```
/*****************************************************************/
/****************      FONCTION  PRESENT          ****************/
/*****************************************************************/


int  present(petit,grand)

   char *petit;
   char *grand;

{
   int i;  /* compteur sur la grande chaine */
   int j;  /* compteur sur la petite chaine */
   int TROUVE;

   i = 0;
   TROUVE = FALSE;
   while (( i < (strlen(grand) - strlen(petit) + 1))
         && (TROUVE == FALSE))
   {
      TROUVE = TRUE;
      j = 0;
      while (( j < (strlen(petit)))
            && (TROUVE == TRUE))
      {
         if (petit[j] != grand[i + j - 1])
         {
            TROUVE = FALSE;
         }
         j = j + 1;
      }
      i = i + 1;
   }
   return(TROUVE);
}   /* fin de la fonction present */
```

```
/*****************************************************************/
/****************        FONCTION  DOM1          *****************/
/*****************************************************************/


dom1(objet,attribut,type,fichier)

   char  objet[13];
   char  *attribut;
   char  type[12];
   FILE  *fichier;

{
   char  domaine[90];
   int   TROUVE;
   char  concatenation[30];
   int   i,j;

   strcpy(concatenation,type);
   strcat(concatenation,objet);
   attribut[0] = '\0';
   fseek(fichier,0,0);
   TROUVE = FALSE;
   while ((TROUVE == FALSE) &&
          (fgets(domaine,90,fichier) != NULL))
   {
      if (present(concatenation,domaine) == TRUE)
      {
         TROUVE = TRUE;
         fgets(domaine,90,fichier);
         fgets(domaine,90,fichier);
         i = 0;
         while (domaine[i] == ' ')
            i = i + 1;
         j = 0;
         while (domaine[i] != ';')
         {
            attribut[j] = domaine[i];
            i = i + 1;
            j = j + 1;
         }
         attribut[j] = '\0';
      }
   }
}   /*  fin de la fonction dom1  */
```

```
/*****************************************************************/
/****************        FONCTION  DOM2          *****************/
/*****************************************************************/

dom2(objet,attribut,type,fichier)

   char  objet[13];
   char  attribut[10][13];
   char  type[12];
   FILE  *fichier;

{
   char  domaine[90];
   int   TROUVE;
   char  concatenation[30];
   int   i,j,k;

   strcpy(concatenation,type);
   strcat(concatenation,objet);
   for ( i = 0 ; i < 10 ; i++ )
       attribut[i][0] = '\0';
   fseek(fichier,0,0);
   TROUVE = FALSE;
   while ((TROUVE == FALSE) &&
          (fgets(domaine,90,fichier) != NULL))
   {
      if (present(concatenation,domaine) == TRUE)
      {
         TROUVE = TRUE;
         fgets(domaine,90,fichier);
         i = 0;
         while (present("DOMAIN\0",domaine) == TRUE)
         {
            fgets(domaine,90,fichier);
            j = 0;
            while (domaine[j] == ' ')
                  j = j + 1;
            k = 0;
            while (domaine[j] != ';')
            {
               attribut[i][k] = domaine[j];
               j = j + 1;
               k = k + 1;
            }
            attribut[i][k] = '\0';
            i = i + 1;
            fgets(domaine,90,fichier);
         }
      }
   }
}   /*  fin de la fonction dom2  */
```

```
/*****************************************************************/
/****************      FONCTION  DOM3          *******************/
/*****************************************************************/


dom3(objet,attribut,type,fichier)

   char  objet[13];
   char  *attribut;
   char  type[12];
   FILE  *fichier;

{
   char  domaine[90];
   int   TROUVE;
   int   FIN;
   char  concatenation[30];
   int   i,j;

   strcpy(concatenation,type);
   strcat(concatenation,objet);
   attribut[0] = '\0';
   fseek(fichier,0,0);
   TROUVE = FALSE;
   while ((TROUVE == FALSE) &&
          (fgets(domaine,90,fichier) != NULL))
   {
      if (present(concatenation,domaine) == TRUE)
      {
         TROUVE = TRUE;
         fgets(domaine,90,fichier);
         fgets(domaine,90,fichier);
         i = 0;
         while (domaine[i] != '\'')
              i = i + 1;
         j = 0;
         i = i + 1;
         FIN = FALSE;
         while ((domaine[i] != '\'') || (FIN == FALSE))
         {
            if ((domaine[i] == '\'') && (domaine[i + 1] == ';'))
                  FIN = TRUE;
            else
            {
               attribut[j] = domaine[i];
               i = i + 1;
               j = j + 1;
            }
         }
         attribut[j] = '\0';
      }
   }
}   /*  fin de la fonction dom3  */
```

```
/*****************************************************************/
/****************        FONCTION DOM4          ******************/
/*****************************************************************/


dom4(objet,tabligne,tabcolonne,type,fichier)

   char  objet[13];
   char  type[12];
   int   tabligne[10];
   int   tabcolonne[10];
   FILE  *fichier;

{
   char  domaine[90];
   int   TROUVE;
   char  concatenation[30];
   int   i,j,k;
   char  s[3];

   strcpy(concatenation,type);
   strcat(concatenation,objet);
   for (i = 0 ; i < 10 ; i++)
   {
      tabligne[i] = 0;
      tabcolonne[i] = 0;
   }
   fseek(fichier,0,0);
   TROUVE = FALSE;
   while ((TROUVE == FALSE) &&
          (fgets(domaine,90,fichier) != NULL))
   {
      if (present(concatenation,domaine) == TRUE)
      {
         TROUVE = TRUE;
         fgets(domaine,90,fichier);
         i = 0;
         while (present("DOMAIN\0",domaine) == TRUE)
         {
            fgets(domaine,90,fichier);
            j = 0;
            while (domaine[j] != '\'')
                j = j + 1;
            j = j + 1;
            k = 0;
            while (domaine[j] != '_')
            {
               s[k] = domaine[j];
               k = k + 1;
               j = j + 1;
            }
```

```
            s[k] = '\0';
            tabligne[i] = atoi(s);
            k = 0;
            j = j + 1;
            while (domaine[j] != '\'')
            {
               s[k] = domaine[j];
               k = k + 1;
               j = j + 1;
            }
            s[k] = '\0';
            tabcolonne[i] = atoi(s);
            i = i + 1;
            fgets(domaine,90,fichier);
         }
      }
   }
}   /*  fin de la fonction dom4  */
```

```
/*****************************************************************/
/****************   FONCTION  CONS_MESSAGE      ****************/
/*****************************************************************/

cons_message(nom_obj)

   char  *nom_obj;

{
   char  m_type[13];
   char  expli_mess[13];
   char  etats[10][13];
   char  m_exped[13];
   char  destin[10][13];
   char  rep_mess[10][13];
   int   i,j;

   /*  traitement du nom du message  */

   TDMScopy(nom_obj,msge.nom_mess,12);

   /*  traitement du type du message  */
   /*  recherche du type  */

   dom1(nom_obj,m_type,"type_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   TDMScopy(m_type,msge.type_mess,12);
```

```
/*  traitement de l'explication du message  */

if (strcmp(m_type,"autre\0") == 0)
   dom1(nom_obj,expli_mess,"explication_\0",DOMAINES);
else expli_mess[0] = '\0';
TDMScopy(expli_mess,msge.explication_mess,12);

/*  traitement des états du message  */
/*  recherche des états  */

dom2(nom_obj,etats,"etat_\0",DOMAINES);

/*  traduction compatible avec TDMS  */

i = 0;
while (etats[i][0] != '\0')
{
   TDMScopy(etats[i],msge.m_etats_poss[i].etats_mess,12);
   i = i + 1;
}
while ( i < 10 )
{
   for ( j = 0 ; j < 12 ; j++)
       msge.m_etats_poss[i].etats_mess[j] = ' ';
   i = i + 1;
}

/*  traitement de l'expéditeur du message  */
/*  recherche de l'expéditeur  */

assoc1(nom_obj,m_exped,"_EXP_\0",EXPEDITEURS);

/*  traduction compatible avec TDMS  */

TDMScopy(m_exped,msge.exped_mess,12);

/*  traitement des destinataires du message */
/*  recherche des destinataires  */

assoc2(nom_obj,destin,10,"_DEST_\0",DESTINATAIRES);

/*  traduction compatible avec TDMS  */

i = 0;
while (destin[i][0] != '\0')
{
   TDMScopy(destin[i],msge.m_destinataires[i].dest_mess,12);
   i = i + 1;
}
while ( i < 10 )
{
   for ( j = 0 ; j < 12 ; j++ )
      msge.m_destinataires[i].dest_mess[j] = ' ';
   i = i + 1;
}
```

```
   /*  traitement des réponses du message  */
   /*  recherche des réponses  */

   assoc2(nom_obj,rep_mess,10,"_REP_\0",REPONSES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (rep_mess[i][0] != '\0')
   {
      TDMScopy(rep_mess[i],msge.reponse_mess[i].nom,12);
      i = i + 1;
   }
   while (i < 10)
   {
      for (j = 0 ; j < 12 ; j++)
        msge.reponse_mess[i].nom[j] = ' ';
      i = i + 1;
   }

   /*  appel de la requête TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&MESS_DESC,&msge);
   test(RET_STAT);
}  /*  fin de la fonction cons_message  */



/*****************************************************************/
/****************   FONCTION  CONS_DOCUMENT     ****************/
/*****************************************************************/


cons_document(nom_obj)

   char  *nom_obj;

{
   char  d_type[13];
   char  expli_doc[13];
   char  etats[10][13];
   char  d_exped[13];
   char  destin[10][13];
   char  rep_doc[10][13];
   int   i,j;


   /*  traitement du nom du document  */

   TDMScopy(nom_obj,doct.nom_doc,12);
```

```
/*  traitement du type du document  */
/*  recherche du type  */

dom1(nom_obj,d_type,"type_\0",DOMAINES);

/*  traduction compatible avec TDMS  */

TDMScopy(d_type,doct.type_doc,12);

/*  traitement de l'explication du document  */

if (strcmp(d_type,"autre\0") == 0)
   dom1(nom_obj,expli_doc,"explication_\0",DOMAINES);
else expli_doc[0] = '\0';
TDMScopy(expli_doc,doct.explication_doc,12);

/*  traitement des états du document  */
/*  recherche des états  */

dom2(nom_obj,etats,"etat_\0",DOMAINES);

/*  traduction compatible avec TDMS  */

i = 0;
while (etats[i][0] != '\0')
{
   TDMScopy(etats[i],doct.d_etats_poss[i].etats_doc,12);
   i = i + 1;
}
while ( i < 10 )
{
   for ( j = 0 ; j < 12 ; j++)
       doct.d_etats_poss[i].etats_doc[j] = ' ';
   i = i + 1;
}

/*  traitement de l'expéditeur du document  */
/*  recherche de l'expéditeur  */

assoc1(nom_obj,d_exped,"_EXP_\0",EXPEDITEURS);

/*  traduction compatible avec TDMS  */

TDMScopy(d_exped,doct.exped_doc,12);

/*  traitement des destinataires du document */
/*  recherche des destinataires  */

assoc2(nom_obj,destin,10,"_DEST_\0",DESTINATAIRES);
```

```
   /*  traduction compatible avec TDMS  */

   i = 0;
   while (destin[i][0] != '\0')
   {
      TDMScopy(destin[i],doct.d_destinataires[i].dest_doc,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++ )
         doct.d_destinataires[i].dest_doc[j] = ' ';
      i = i + 1;
   }

   /*  traitement des réponses du document  */
   /*  recherche des réponses  */

   assoc2(nom_obj,rep_doc,10,"_REP_\0",REPONSES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (rep_doc[i][0] != '\0')
   {
      TDMScopy(rep_doc[i],doct.reponse_doc[i].nom,12);
      i = i + 1;
   }
   while (i < 10)
   {
      for (j = 0 ; j < 12 ; j++)
        doct.reponse_doc[i].nom[j] = ' ';
      i = i + 1;
   }

   /*  appel de la requete TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOC_DESC,&doct);
   test(RET_STAT);
}  /*  fin de la fonction cons_document  */
```

```
/*****************************************************************/
/****************   FONCTION  CONS_DOSSIER      ******************/
/*****************************************************************/


cons_dossier(nom_obj)

   char  *nom_obj;

{
   char  etats[10][13];
   char  do_exped[13];
   char  destin[10][13];
   char constituants[20][13];
   int   i,j;


   /*  traitement du nom du dossier  */

   TDMScopy(nom_obj,doss.nom_dossier,12);

   /*  traitement des états du dossier  */
   /*  recherche des états  */

   dom2(nom_obj,etats,"etat_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (etats[i][0] != '\0')
   {
      TDMScopy(etats[i],doss.do_etats_poss[i].etats_doss,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++)
          doss.do_etats_poss[i].etats_doss[j] = ' ';
      i = i + 1;
   }

   /*  traitement de l'expéditeur du dossier  */
   /*  recherche de l'expéditeur  */

   assoc1(nom_obj,do_exped,"_EXP_\0",EXPEDITEURS);

   /*  traduction compatible avec TDMS  */

   TDMScopy(do_exped,doss.exped_doss,12);
```

```
   /*  traitement des destinataires du dossier */
   /*  recherche des destinataires  */

   assoc2(nom_obj,destin,10,"_DEST_\0",DESTINATAIRES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (destin[i][0] != '\0')
   {
      TDMScopy(destin[i],doss.do_destinataires[i].dest_doss,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++ )
         doss.do_destinataires[i].dest_doss[j] = ' ';
      i = i + 1;
   }

   /*  traitement des constituants du dossier  */
   /*  recherche des constituants  */

   assoc2(nom_obj,constituants,20,"_CONS_\0",CONSTITUANTS);

   /*  traduction compatible avec tdms  */

   i = 0;
   while (constituants[i][0] != '\0')
   {
      TDMScopy(constituants[i],doss.constitue_doss[i].const_doss,
                                                          12);
      i = i + 1;
   }
   while (i < 20)
   {
      for ( j = 0 ; j < 12 ; j++ )
           doss.constitue_doss[i].const_doss[j] = ' ';
      i = i + 1;
   }

   /*  appel de la requete TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&DOSS_DESC,&doss);
   test(RET_STAT);
}  /*  fin de la fonction cons_dossier  */
```

```
/*****************************************************************/
/****************   FONCTION  CONS_FICHIER      ****************/
/*****************************************************************/


cons_fichier(nom_obj)

   char  *nom_obj;

{
   char  etats[10][13];
   char  constituant[13];
   int   i,j;


   /*  traitement du nom du fichier  */

   TDMScopy(nom_obj,fich.nom_fichier,12);

   /*  traitement des états du fichier  */
   /*  recherche des états  */

   dom2(nom_obj,etats,"etat_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (etats[i][0] != '\0')
   {
      TDMScopy(etats[i],fich.fi_etats_poss[i].etats_fich,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++)
          fich.fi_etats_poss[i].etats_fich[j] = ' ';
      i = i + 1;
   }

   /*  traitement du constituant du fichier  */
   /*  recherche du constituant  */

   assoc1(nom_obj,constituant,"_CONS_\0",CONSTITUANTS);

   /*  traduction compatible avec TDMS  */

   TDMScopy(constituant,fich.constitue_fich.const_fich,12);

   /*  appel de la requete TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FICH_DESC,&fich);
   test(RET_STAT);
}  /*  fin de la fonction cons_fichier  */
```

```
/*****************************************************************/
/****************   FONCTION  CONS_PILE          *****************/
/*****************************************************************/


cons_pile(nom_obj)

   char  *nom_obj;

{
   char  etats[10][13];
   int   i,j;


   /*  traitement du nom de la pile  */

   TDMScopy(nom_obj,tas.nom_pile,12);

   /*  traitement des états de la pile  */
   /*  recherche des états  */

   dom2(nom_obj,etats,"etat_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (etats[i][0] != '\0')
   {
      TDMScopy(etats[i],tas.pi_etats_poss[i].etats_pile,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++)
          tas.pi_etats_poss[i].etats_pile[j] = ' ';
      i = i + 1;
   }

   /*  appel de la requete TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PILE_DESC,&tas);
   test(RET_STAT);
}  /*  fin de la fonction cons_pile  */
```

```
/*****************************************************************/
/****************   FONCTION  CONS_FORMULAIRE   ******************/
/*****************************************************************/


cons_formulaire(nom_obj)

   char  *nom_obj;

{
   char  f_type[11];
   char  etats[10][13];
   char  f_exped[13];
   char  destin[10][13];
   char  rep_form[10][13];
   char  decomp_form[20][13];
   int   i,j;


   /*  traitement du nom du formulaire  */

   TDMScopy(nom_obj,form.titre_form,12);

   /*  traitement du type du formulaire  */
   /*  recherche du type  */

   dom1(nom_obj,f_type,"type_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   TDMScopy(f_type,form.type_form,10);

   /*  traitement des états du formulaire  */
   /*  recherche des états  */

   dom2(nom_obj,etats,"etat_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (etats[i][0] != '\0')
   {
      TDMScopy(etats[i],form.f_etats_poss[i].etats_form,12);
      i = i + 1;
   }
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++)
          form.f_etats_poss[i].etats_form[j] = ' ';
      i = i + 1;
   }
```

```
/*  traitement de l'expéditeur du formulaire  */
/*  recherche de l'expéditeur  */

assoc1(nom_obj,f_exped,"_EXP_\0",EXPEDITEURS);

/*  traduction compatible avec TDMS  */

TDMScopy(f_exped,form.exped_form,12);

/*  traitement des destinataires du formulaire */
/*  recherche des destinataires  */

assoc2(nom_obj,destin,10,"_DEST_\0",DESTINATAIRES);

/*  traduction compatible avec TDMS  */

i = 0;
while (destin[i][0] != '\0')
{
   TDMScopy(destin[i],form.f_destinataires[i].dest_form,12);
   i = i + 1;
}
while ( i < 10 )
{
   for ( j = 0 ; j < 12 ; j++ )
      form.f_destinataires[i].dest_form[j] = ' ';
   i = i + 1;
}

/*  traitement des reponses du formulaire  */
/*  recherche des reponses  */

assoc2(nom_obj,rep_form,10,"_REP_\0",REPONSES);

/*  traduction compatible avec TDMS  */

i = 0;
while (rep_form[i][0] != '\0')
{
   TDMScopy(rep_form[i],form.reponse_form[i].nom,12);
   i = i + 1;
}
while (i < 10)
{
   for (j = 0 ; j < 12 ; j++)
     form.reponse_form[i].nom[j] = ' ';
   i = i + 1;
}

/*  appel de la requete TDMS  */

RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&FORM_DESC,&form);
test(RET_STAT);
```

```
   /*  recherche des parties du formulaire  */

   assoc2(nom_obj,decomp_form,20,"_COMP_\0",PARTIES);

   /*  consultation des parties  */

   i = 0;
   while (decomp_form[i][0] != '\0')
   {
       cons_partie(decomp_form[i]);
       i = i + 1;
   }

}  /*  fin de la fonction cons_formulaire  */




/*****************************************************************/
/***************   PONCTION  CONS_PARTIE       ***************/
/*****************************************************************/


cons_partie(nom_part)

   char  *nom_part;

{
   char  etats[10][13];
   int   tabligne[10];
   int   tabcolonne[10];
   char  tabtxt[20][13];
   char  tabvar[20][13];
   int   i,j;


   /*  traitement du nom de la partie  */

   TDMScopy(nom_part,part.nom_partie,12);

   /*  traitement des états de la partie  */
   /*  recherche des états  */

   dom2(nom_part,etats,"etat_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while (etats[i][0] != '\0')
   {
      TDMScopy(etats[i],part.p_etats_poss[i].etats_partie,12);
      i = i + 1;
   }
```

```
   while ( i < 10 )
   {
      for ( j = 0 ; j < 12 ; j++)
          part.p_etats_poss[i].etats_partie[j] = ' ';
      i = i + 1;
   }

   /*  traitement de l'emplacement des parties  */
   /*  recherche de l'emplacement  */

   dom4(nom_part,tabligne,tabcolonne,"emplacement_\0",DOMAINES);

   /*  traduction compatible avec TDMS  */

   i = 0;
   while  ((tabcolonne[i] != 0) || (tabligne[i] != 0))
   {
      part.emplac_partie[i].ligne_part = tabligne[i];
      part.emplac_partie[i].colonne_part = tabcolonne[i];
      i = i + 1;
   }

   /*  appel de la requête TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&PART_DESC,&part);
   test(RET_STAT);

   /*  recherche des éléments de texte de la partie  */

   assoc2(nom_part,tabtxt,20,"_COMT_\0",SQUELETTES);

   /*  recherche des éléments variables de la partie  */

   assoc2(nom_part,tabvar,20,"_COMV_\0",VARIABLES);

   /*  consultation des éléments de texte  */

   i = 0;
   while (tabtxt[i][0] != '\0')
   {
      cons_texte(tabtxt[i]);
      i = i + 1;
   }

   /*  consultation des éléments variables de la partie  */

   i = 0;
   while (tabvar[i][0] != '\0')
   {
      cons_variable(tabvar[i]);
      i = i + 1;
   }

}  /*  fin de la fonction cons_partie  */
```

```
/*****************************************************************/
/****************    FONCTION  CONS_TEXTE       ****************/
/*****************************************************************/


cons_texte(nom)

   char  *nom;

{
   char  txt_format[31];
   char  txt_contenu[31];
   int   tabligne[10];
   int   tabcolonne[10];
   int   i,j;

   /*  traitement du nom de l'élément du squelette textuel  */

   TDMScopy(nom,elt_txt.nom_texte,12);

   /*  traitement du format de l'élément de texte  */

   dom3(nom,txt_format,"format_\0",DOMAINES);
   TDMScopy(txt_format,elt_txt.format,30);

   /*  traitement du contenu de l'élément de texte  */

   dom3(nom,txt_contenu,"contenu_\0",DOMAINES);
   TDMScopy(txt_contenu,elt_txt.contenu,30);

   /*  traitement de l'emplacement de l'élément de texte  */

   dom4(nom,tabligne,tabcolonne,"emplacement_\0",DOMAINES);
   i = 0;
   while ((tabcolonne[i] != 0) || (tabligne[i] != 0))
   {
      elt_txt.emplac_texte[i].ligne_txt = tabligne[i];
      elt_txt.emplac_texte[i].colonne_txt = tabcolonne[i];
      i = i + 1;
   }

   /*  affichage de la requête TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&TEXTE_DESC,
                          &elt_txt);
   test(RET_STAT);

}   /*  fin de cons_texte  */
```

```
/*****************************************************************/
/****************    FONCTION  CONS_VARIABLE    ****************/
/*****************************************************************/

cons_variable(nom)

   char  *nom;

{
   char  var_format[31];
   char  var_defaut[31];
   char  correspondant[13];
   int   tabligne[10];
   int   tabcolonne[10];
   int   i,j;

   /*  traitement du nom de l'élément variable  */

   TDMScopy(nom,elt_var.nom_var,12);

   /*  traitement du format de l'élément variable  */

   dom3(nom,var_format,"format_\0",DOMAINES);
   TDMScopy(var_format,elt_var.format,30);

   /*  traitement de la valeur par défaut de l'élément
       variable  */

   dom3(nom,var_defaut,"val_def_\0",DOMAINES);
   TDMScopy(var_defaut,elt_var.val_def,30);

   /*  traitement de l'emplacement de l'élément variable  */

   dom4(nom,tabligne,tabcolonne,"emplacement_\0",DOMAINES);
   i = 0;
   while ((tabcolonne[i] != 0) || (tabligne[i] != 0))
   {
      elt_var.emplac_var[i].ligne_var = tabligne[i];
      elt_var.emplac_var[i].colonne_var = tabcolonne[i];
      i = i + 1;
   }

   /*  traitement de l'élément textuel correspondant  */

   assoc1(nom,correspondant,"_CORR_\0",CORRESPONDANTS);
   TDMScopy(correspondant,elt_var.txt_cor,12);

   /*  affichage de la requête TDMS  */

   RET_STAT = TSS$REQUEST(&CHANNEL,&LIBRARY_ID,&VAR_DESC,
                          &elt_var);
   test(RET_STAT);

}   /*  fin de cons_var  */
```

MANUEL D'UTILISATION DES PROGRAMMES

1. Comment exécuter les programmes ?

Le programme qui permet de décrire les objets de bureau s'appelle "BUREAU".
Celui qui permet de consulter les objets de bureau décrits s'appelle "CONSULTATION".
On peut exécuter ces programmes par la commande "RUN":

```
RUN BUREAU
RUN CONSULTATION
```

2. Comment se positionner à différents endroits dans les écrans ?

La touche TAB permet de passer au champ suivant.

Les flèches du clavier auxiliaire permettent de passer d'un champ à l'autre dans une zone répétitive, selon le sens des flèches. La touche TAB permet également de se positionner dans ces zones répétitives et doit être utilisée pour en sortir.

La touche RETURN permet de passer à l'écran suivant.
!!! Faites bien attention de ne pas appuyer sur cette touche en cours de remplissage de l'écran sinon vous passeriez à l'écran suivant sans avoir pu compléter le premier !!!

3. Comment remplir les zones des écrans ?

L'utilisateur devra remplir son écran en minuscules. Cette convention a été prise pour une question de lisibilité lors du passage en DSL:

- le texte DSL est écrit en majuscules;
- les informations fournies par l'utilisateur sont écrites en minuscules.

Le gestionnaire d'écran TDMS permet de fournir des messages et des écrans d'aide à l'utilisateur.
Si l'on appuie une fois sur la touche PF2 (deuxième touche supérieure gauche du clavier auxiliaire de droite), un message d'aide s'inscrit dans le bas de l'écran.
Si l'on appuie deux fois sur cette touche PF2, un écran d'aide s'affiche au terminal. Pour revenir à l'écran à remplir, il suffit d'appuyer une fois de plus sur cette touche.

Le gestionnaire d'écran TDMS permet à l'utilisateur d'apporter des modifications à l'écran qu'il est en train de remplir. Pour ce, il doit appuyer sur la touche PF1 (première touche supérieure gauche du clavier auxiliaire) et ensuite sur la touche S. Cette opération a pour effet de remettre le curseur au début de l'écran. L'utilisateur peut alors se positionner sur les différents champs qu'il désire modifier.

4. Comment enchainer les programmes ?

Les programmes doivent s'enchainer comme suit:

- Le programme "BUREAU" permet de décrire les objets informationnels du bureau.
- Les commandes de DSL-SPEC permettent la mise dans une base de données des spécifications.
- Le programme "CONSULTATION" permet de consulter les objets informationnels décrits

La mise dans la base de données des spécifications DSL se fera à partir d'un fichier de commandes préalablement préparé: COMMANDES.DSA. La session des commandes va se dérouler à l'écran et le clavier de l'utilisateur sera bloqué jusqu'à la terminaison de la dernière commande du fichier.
Pour invoquer DSL-SPEC dans ce mode, il suffit de donner la commande:

```
$ DSAMLT
```

Le logiciel IDA commence la session en affichant ensuite les lignes suivantes:

```
DSL-SPEC à partir d'un fichier de commandes
    Version anglaise de DSL

Command-file-name: ( 1 )
```

L'utilisateur doit spécifier le nom du fichier contenant ses commandes DSL-SPEC en (1).

L'enchainement des programmes se fera donc par la session de commandes suivantes:

```
RUN BUREAU  ( exécution du programme BUREAU )

INITDB DATABASE  ( initialisation de la base de données )

DSAMLT  ( exécution du fichier des commandes DSL )

  Command-file-name: COMMANDES.DSA

RUN CONSULTATION  (exécution du programme CONSULTATION )
```

BIBLIOGRAPHIE

[IDA] Atelier logiciel IDA d'aide à la conception de systèmes d'information.
Manuel de référence: DSL-SPEC.
F.N.D.P. Namur.

[KERN] B.W. KERNIGHAN et D.M. RITCHIE.
traduit de l'anglais par T. BUFFENOIR.
Le langage C.
MASSON, 1984.

[TDMS 1] DIGITAL SOFTWARE.
VAX-11 TDMS.
Summary description.

[TDMS 2] DIGITAL SOFTWARE.
VAX-11 TDMS.
Forms manual.

[TDMS 3] DIGITAL SOFTWARE.
VAX-11 TDMS.
Request manual.

[TDMS 4] DIGITAL SOFTWARE.
VAX-11 TDMS.
Programming manual.

[TDMS 5] DIGITAL SOFTWARE.
VAX-11 TDMS.
Sample application.